STANFORD RESEARCH INSTITUTE
MENLO PARK, CALIFORNIA
SRI

April 1961

Technical Report 1

AUTOMATIC FAULT DETECTION IN COMBINATIONAL SWITCHING NETWORKS

Prepared for:

JET PROPULSION LABORATORY
4800 OAK GROVE AVENUE
PASADENA, CALIFORNIA

CONTRACT M-44501 UNDER
NASA CONTRACT NASw-8

By: William H. Kautz

SRI Project No. 3196

Approved:

J. R. ANDERSON, MANAGER COMPUTER TECHNIQUES LABORATORY

J. D. NOE, DIRECTOR ENGINEERING SCIENCES DIVISION

Copy No.....99

ABSTRACT

This report is concerned with the logical design of economical combinational switching networks which contain sufficient redundancy so that the presence of any single fault can be detected.

It is well known that (within broad limits) any isolated fault in a single-output network may be detected with about 2:1 redundancy, through duplication of the irredundant network, and the addition of a comparator gate. In the procedures to be described, a reduction of the redundancy ratio below 2:1 is shown to be usually possible, by (1) taking advantage of any inherent logical redundancy in the switching function or functions which describe the terminal behavior of the network; (2) making use of any structural redundancy present in the irredundant version of the network; (3) providing for the detection of only those faults considered to be at all likely to occur, rather than "all" faults categorically; (4) where possible, allowing some faults to be detected with a delay, rather than at the first occurrence of an erroneous output; and (5) application of certain principles of error-detecting codes. A similar reduction of the redundancy ratio is possible for multi-output networks.

Branch-type networks—*e.g.*, those made up of relay contacts or cryotrons instead of gate-type elements—are also considered. If the designer is free to use some non-branch-type elements in the detection circuitry, the same low redundancy ratios achievable in gate-type circuitry can be obtained here.

Finally, several examples in application areas of importance in conventional digital systems are discussed, including some comparators, calculating networks, code converters, and decoding trees.

CONTENTS

ILLUSTRATIONS

AUTOMATIC FAULT DETECTION IN COMBINATIONAL SWITCHING NETWORKS

I INTRODUCTION

The high reliability needed in today's complex digital systems requires the use of the best available components, design practice, interconnection techniques and testing procedure, along with a heavy reliance on duplication, spares, and "design for maintenance." In applications where space, weight, and power requirements are at a premium, lower reliability must be accepted unless more shrewd design procedures can be developed to allow a system to be operable in the face of defective components. Future reliability requirements will undoubtedly be even more severe, as a result of further increases in system complexity. It is unlikely that component, assembly, and interconnection improvements alone will keep pace with these requirements.

Thus, logical and system designers are faced with the problem of modifying and extending available design procedures to make the best use of a modest amount of circuit redundancy, for the purpose of improving the reliability of the over-all system beyond that achievable with the best irredundant design. Adequate techniques are not presently available for this purpose.

It has been known for some time that arbitrarily high reliability can be achieved through the application of a sufficient degree of redundancy.[1,2]* The truth of this observation, which is based on reasonable assumptions, is a reflection of the fact that, although the likelihood of failures tends to increase approximately linearly with the total number of elements used, the number of faults observable at the terminals of the system can be made to decrease at an exponential rate if redundant elements are properly incorporated. While the offered proofs are constructive, they require very large redundancy cost ratios before the exponential decrease overcomes the linear increase, for typical component

* References are listed at the end of the report.

failure probabilities. However, exploratory work with specific digital networks of various types suggests that substantial improvements in reliability may be possible with relatively small redundancy ratios. This report describes the results of part of a study of methods for achieving such improvements. This reduction is achieved by (1) applying the redundancy at the proper level (*i.e.*, selecting the level of the component, or the logical or storage element, or the network, or the subsystem, etc.), (2) taking advantage of the particular operation being performed, (3) utilizing the structure of the irredundant network or system, which may contain some inherent redundancy, (4) fault-checking only the types of fault which may be reasonably expected to occur, rather than "all" faults, and (5) an application of some of the principles of error-checking codes.

To make the best use of a limited amount of circuit or logical redundancy, the designer must first decide just how the extra equipment is to be allocated between the various aspects of reliable behavior: fault detection, fault location, and fault correction (*i.e.*, repair, including switchover to a spare). To these should be added a fourth, which we shall designate *fault-masking*, to indicate the operation of compensating for a fault without the benefit of knowing its location or even its nature. For example, fault masking is involved in the replacement of a single diode or contact by a series-parallel "quad," as shown in Figs. 1(a) and (b), or the replacement of an r-input AND- or OR-gate by the configurations in Figs. 1(c) and (d). Any single diode or contact may become shorted or open without affecting the digital behavior of the redundant arrangement as a whole. However, the actual identification of the faulty diode or element in the set would require that the circuit be disconnected to the point where the diodes or gates can be individually tested.

The proper allocation of added redundancy between detection, location, correction, and masking depends on the nature of the application of the system and is beyond the concern of this study, except to the extent that one of these objectives (*e.g.*, correction) might be obtained through the mechanism of others (masking plus detection). Detection is usually the least costly, and correction the most costly, with masking in between. The cost of fault location depends strongly on how small is the part of the circuit to which the location of the fault is to be narrowed.

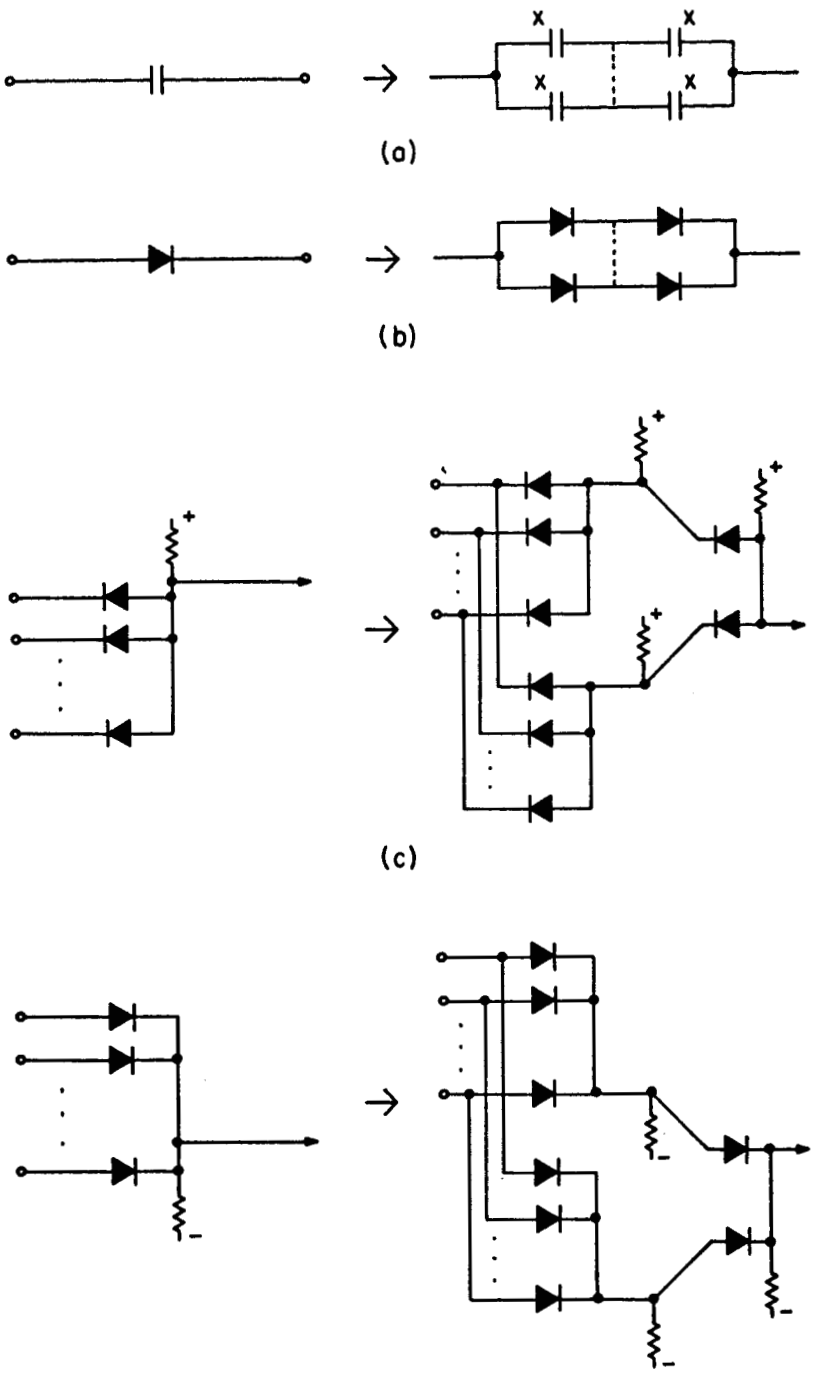


FIG. 1

FAULT-MASKING TECHNIQUES FOR CIRCUIT COMPONENTS AND GATES

This report is concerned with techniques for modifying the design of combinational digital networks so that any single isolated fault is *detected*. The limitation to single faults does not, of course, exclude the detection of most double and other multiple faults, but assumes that the network is already sufficiently reliable so that multiple faults are much less likely to occur than single faults. Such an assumption allows the analysis to free itself of all probabilistic considerations, and is also a reflection of the primary purpose of the fault-detecting feature, which is to provide indication of the *first* failure.

Despite an extensive literature on the use of redundancy for fault-masking and fault-correction,* very little has been published in the technical literature on the subject of fault detection.

It is known that *all* single faults in a single-output combinational network can be detected in an arrangement having a redundancy ratio of a little more than 2:1. We show in the following sections that the ratio required for the most likely types of faults may be considerably less than 2:1, however, and tends to become smaller as the size of the network is increased.

Multiple-output networks also require proportionally less redundancy, even for the detection of all faults. Branch-type networks (*e.g.*, those made up of relay contacts) are more difficult to handle, because of the completely bilateral nature of the basic switching element. If the designer is free to use some non-branch-type elements in the detection circuitry, however, the same redundancy ratios achievable in gate-type circuitry can be obtained. Several application areas common to conventional digital systems are discussed as examples in the sections to follow.

* See, *e.g.*, the bibliography attached to Quarterly Letter Report 2, 14 October 1960, and to be included in the Final Report of this project.

II SINGLE-OUTPUT GATE-TYPE NETWORKS

A. NETWORK REDUNDANCY

We consider first switching networks composed of *gate-type* logical elements such as AND, OR, NOT, NOR, etc. gates, as opposed to *branch-type* elements, such as relay contacts, mechanical switches, cryotrons, a certain class of transistor switches, etc. Branch-type circuits will be discussed in Sec. IV. The single-output gate-type network N of Fig. 2(a) is to be replaced with a redundant version N_1 having two outputs, as in Fig. 2(b). One of these two outputs is describable by the switching function f, as was the output of the irredundant network, and the other output labelled e is to have the value 1 when and only when a single fault occurs within the network N_1. (Precisely what constitutes a "fault" will be considered shortly.)

Since the output e must be sensitive to faults in *all* elements in N_1, that particular element which generates the output f must also contribute to the output e. Thus, the network N_1 must have the form shown in Fig. 2(c), in which C is the portion of the network which forms e from f, and from the input variables, which are combined into two intermediate signals g_0 and g_1 in accordance with an arbitrary dependence of e upon f:

$$e = g_0\bar{f} + g_1 f \quad .$$

Here, g_0 and g_1 are functions of the input variables only.

To maintain e at the value $\underline{0}$ in the absence of faults, it is necessary that $g_0 = 0$ when $f = 0$, and $g_1 = 0$ when $f = 1$. In the presence of any fault which causes a $\underline{1}$ f-output to become a $\underline{0}$, g_0 must take on the value $\underline{1}$ under the same input conditions. Similarly, for every fault which causes a $\underline{0}$ f output to become a $\underline{1}$, g_1 must take on the value $\underline{1}$. To satisfy these conditions, it is sufficient to select g_0 and g_1 according to the simple condition

$$g_0 = \bar{g}_1 = f$$

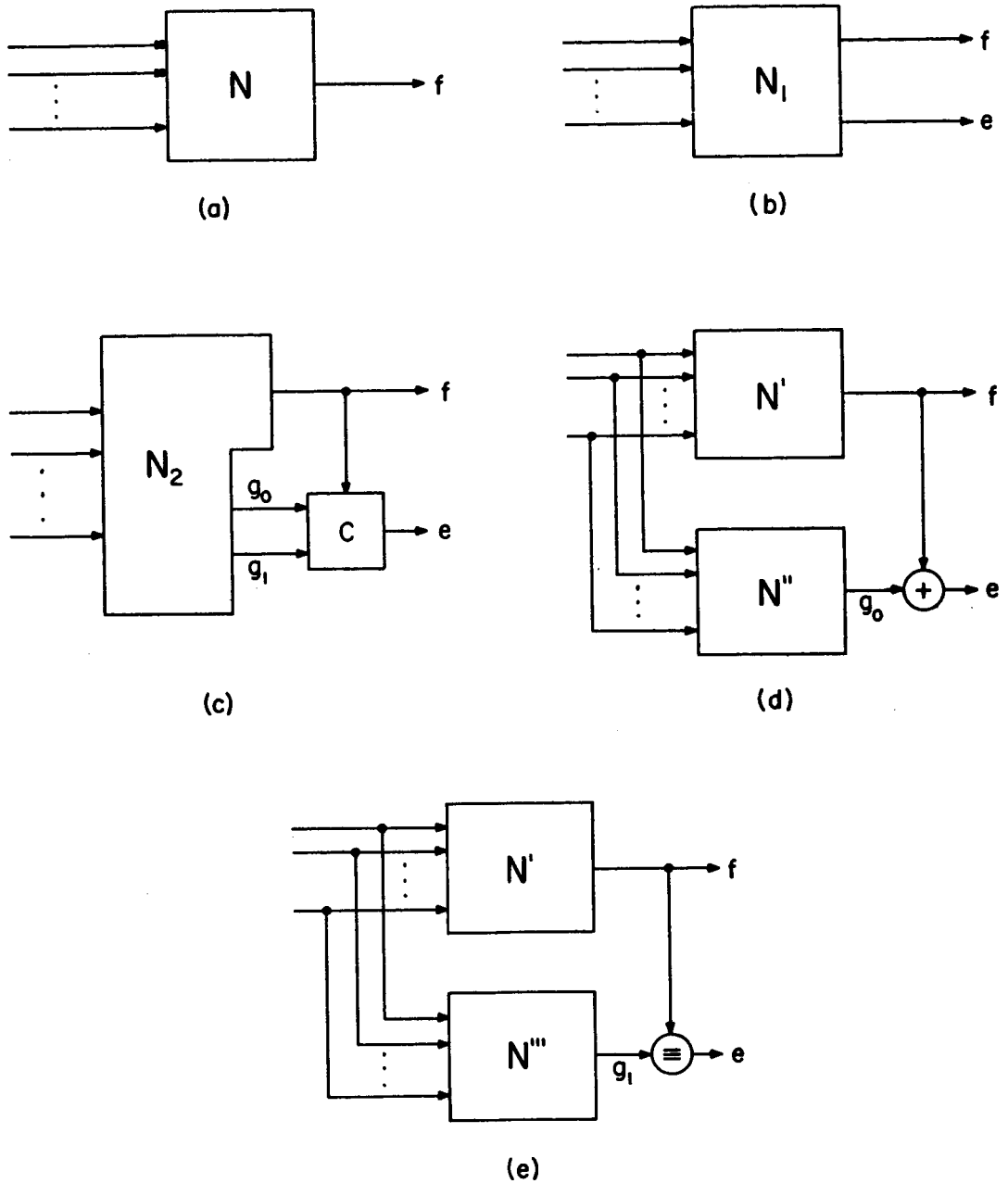


FIG. 2

FAULT-DETECTING SINGLE-OUTPUT NETWORK CONFIGURATIONS

and to form g_0 independent of the f-output, as shown in Fig. 2(d). The error output then becomes

$$e = g_0\bar{f} + \bar{g}_0 f$$

$$e = f \oplus g_0$$

where the symbol "$\oplus$" designates "exclusive OR." It is clear that any fault in the network N' (which we may identify with N for the moment) which causes an error in f for some input condition, will cause e to have the value $\underline{1}$ for that same input condition. By symmetry, any fault in the network N'' (which may be a duplicate of N) will also yield an error output, even though the network output f remains correct. The redundancy (cost) ratio required for this degree of fault detection is a little more than 2:1.

An obvious alternate checking network N''' results if $g_1 = \bar{f}$ is formed instead of $g_0 = f$, in which case the comparison organ should be replaced by a biequivalence gate:

$$e = fg_1 + \bar{f}\bar{g}_1$$

$$e = (f \equiv g_1)$$

as shown in Fig. 2(e).

Both of these alternative configurations provide for the detection of all faults in the entire network N_1, except (1) certain short circuits between the upper and lower subnetworks, and (2) certain failures in the comparison organ C. The first of these possibilities can normally be avoided through physical separation of the two networks, with provision as necessary for separate or near-separate input signal sources. Susceptibility to failures in C is basic: so long as the signal e is represented irredundantly—that is, on a single "wire"—it will always be subject to error due to some kinds of faults in the element producing it. If desired, fault-detection, fault-masking, or fault-correction procedures may be separately applied to the e output—for example, in the manner described by von Neumann for overcoming the critical role of his "majority element."[1] Clearly, however, nothing can be done in the design of the network N_1 itself to circumvent this inherent limitation.

B. TYPES OF FAULTS

The very wide range of types and numbers of faults which are detected by the configurations of Figs. 2(d) and (e) suggests that a lower redundancy ratio may be sufficient to detect only single faults, and those faults which may be considered at all likely to occur. To this end, it will be convenient to distinguish between four classes of fault in gate-type switching networks. These will be described with reference to the examples shown in Fig. 3: an AND-gate, an OR-gate, and a NOR element. Application to other gate-type elements is direct.

(1) *Type I*—ineffective element input: In this type of fault a single input to a logical element loses all of its influence over the behavior of the element, so that the element behaves as if that input were not present. In a rectifier gate, this type of fault would arise from an open-circuited rectifier.

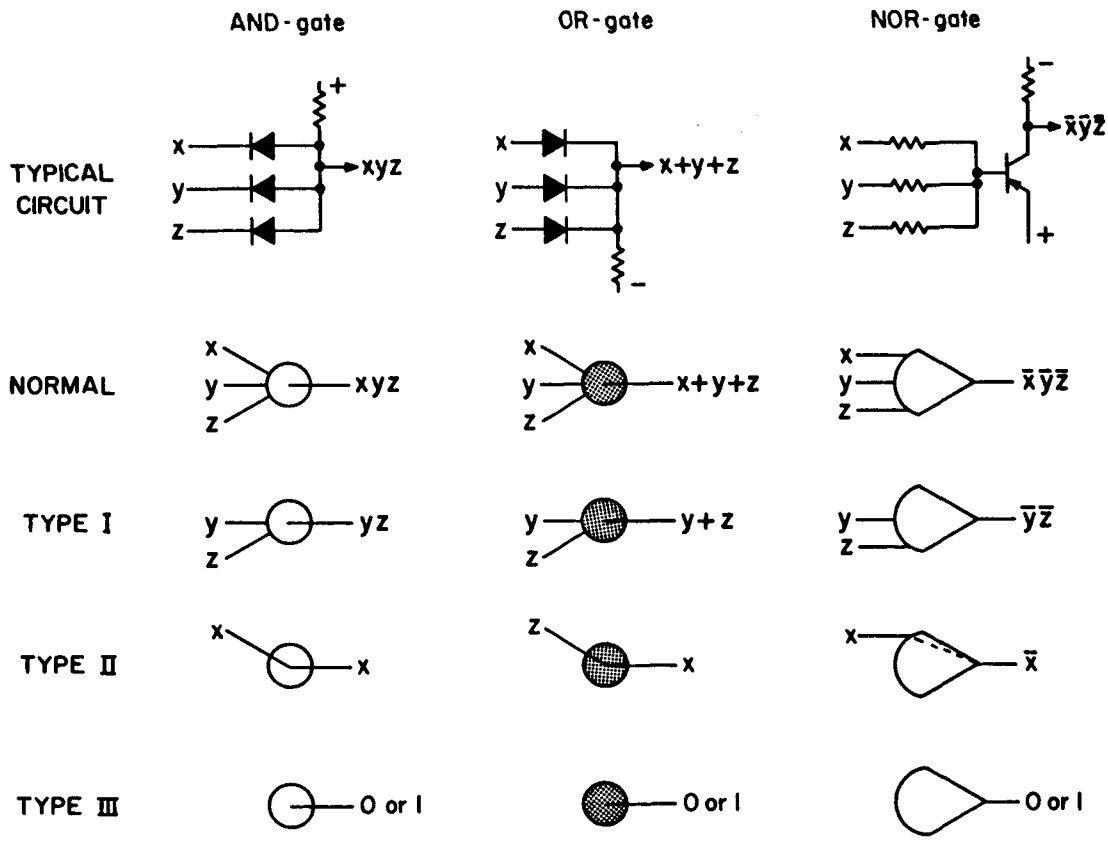


FIG. 3

EFFECTS OF VARIOUS TYPES OF FAULTS ON TYPICAL GATE ELEMENTS

(2) *Type II*—dominant element input: Here a single input to a logical element determines the output value; that is, the element output depends on only one of its inputs, and all other element inputs are ineffective. In a rectifier gate, this type of fault roughly corresponds to a short-circuited rectifier.

(3) *Type III*—dominant element output: Here the element output becomes completely independent of all inputs, and takes on the value 0 or 1 for all input combinations. In a typical NOR circuit, this type of fault might result from a defective transistor.

All other less likely types of faults are classified as Type IV, including more complex distortions of the normal input-output relations of the elements, multiple-element effects due to extreme loading, and random inter-element short circuits within a network. It is felt that in most switching circuits, Type IV faults are considerably less likely than the first three types, and their likelihood can usually be further minimized by knowledgeable circuit design.

In the presence of certain component short circuits which give rise to Type I and II faults (*e.g.*, a shorted rectifier in an AND-gate), it is possible for the effect of such a single fault to be propagated to other gates which share the same input signals. In such a case, the single component fault may appear in the network as a multiple fault, even in cases where the erroneous gate outputs can normally take on independent values.

There are two ways in which this anomaly can occur. If the offensive gate input is driven by the output of another gate internal to the network, then this component failure in the load gate really has the same effect as a Type III failure in the driving gate, and should be interpreted as such. If the offensive gate input is driven by an input signal to the network, then the component failure is tantamount to an unintended change in the state of the input variables. This input change would actually occur only if the source of the input variables is not sufficiently strong to maintain the signal value of the inputs in the presence of the additional loading. If sufficiently serious, such a change can be avoided either by increasing the driving capability of the source signals, or by introducing some degree of isolation between the gate loads on the input signal line (*e.g.*, a small resistor or a fuse in series with rectifiers), or both. It can be

circumvented by applying redundancy at a higher level over the set of input variables, in a manner to be described in Sec. III. In actual practice the entire possibility may be quite negligible if it can be assumed that such Type I and II faults are much more likely due to gradual deterioration than to downright short circuits in the components. In such a case, the input loading is a second-order effect in comparison with the local effect of improper gate operation, so that the fault would first be detected on the latter basis.

C. AND-OR NETWORKS

Consider first single-output AND-OR networks—that is, those composed entirely of AND and OR elements. Such networks enjoy a wide usage, particularly in two-level AND/OR and OR/AND configurations,* and minimization procedures have been available for such irredundant two-level networks for some time, even in the multiple-output case.[3,4]

From the AND-gate column of Fig. 3 it is apparent that in the presence of any fault of Type I or II, any AND-gate output function *is implied by* the fault-free gate output function. That is, the effect of any such fault is to cause the gate output to take on the value 1 when it should be 0 for some input combinations, but the fault will never change a 1 output to a 0. Since there is no possibility of inversion in an AND-OR network, the same is true of the network output in the presence of AND-gate failures—that is, the erroneous output function is always implied by the correct output function. Thus, all Type I and II AND-gate failures may be detected through a duplication of the original network, as in Fig. 2(d) with $g_0 = f$, but now the comparison organ C is somewhat simpler:

$$e = \bar{g}_0 f \quad .$$

Alternatively, we may select $g_1 = \bar{f}$ for the second network [Fig. 2(e)], and use a simple AND-element for the comparison organ:

$$e = g_1 f \quad .$$

This arrangement, shown in Fig. 4(a), is protected against Type I and II failures not only in the upper and lower networks, but in the comparison

* We use the diagonal "/" to indicate two-level networks, such as in the expression AND/OR for a level of AND-gates followed by an output OR-gate. The notation "AND-OR" will be used for networks of AND and OR elements with an unspecified number of levels.

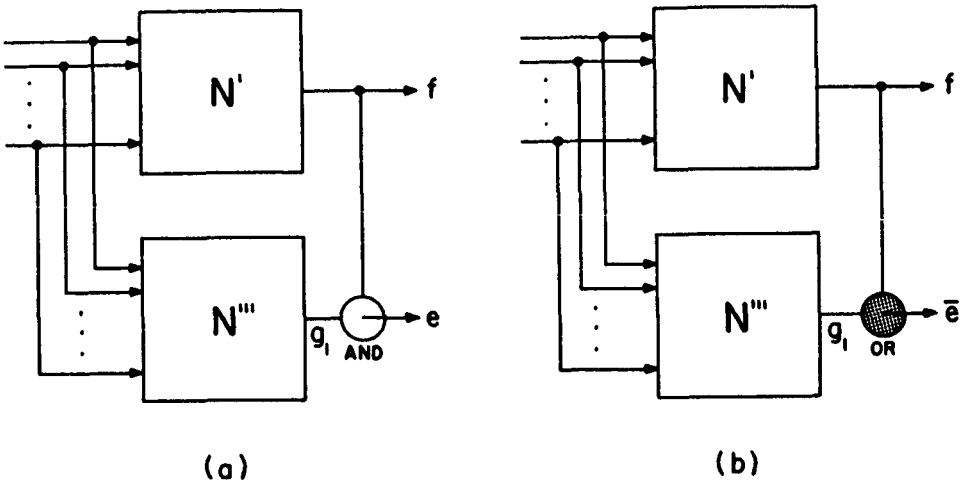


FIG. 4

POSSIBLE NETWORK CONFIGURATIONS FOR DETECTING TYPE I AND II FAULTS IN (a) AND-GATES ONLY AND (b) OR-GATES ONLY

AND-gate as well: such failures can only result in either $e = g_1 = \overline{f}$, or $e = f$, and are detected either immediately or at the next change of value of f.

The dual situation maintains for OR-gate failures of Types I and II. From the OR-gate column of Fig. 3, it is apparent that faults in these elements result in gate outputs, hence network outputs, which *imply* the proper output, and hence result in some 1's being replaced by 0's in the pattern of network outputs. No 0's are replaced by 1's, however, so the selection $g_0 = f$ for the added network allows e to be simplified to

$$e = g_0\overline{f} \quad .$$

The selection $g_1 = \overline{f}$ yields

$$e = \overline{g}_1\overline{f}$$

or, if complementary error detection is permitted,

$$\overline{e} = g_1 + f \quad .$$

That is, the $\overline{e}$ output is maintained at the value $\underline{1}$ until a fault occurs, when it takes on the value $\underline{0}$. This network is shown in Fig. 4(b). Again, all single Type I and II faults are detected, whether in the upper or lower networks, or in the comparison OR-gate.

In general, the complexity of an AND-OR realization of $\overline{f}$ is the same as that of f, since either can be obtained from the other by interchanging AND-gates and OR-gates, and complementing all input variables. If the upper and lower networks are restricted to have only two levels in either AND/OR or OR/AND fashion, but both the same, the realization of $\overline{f}$ may be simpler or more complex than the realization of f. Actually, the costs of the two are comparable for the majority of switching functions.

The configurations just described also detect some Type III faults in all but the last (output-most) level of the network—namely, those in which AND-gate outputs tend toward $\underline{1}$ in the first case, or OR-gate outputs tend toward $\underline{0}$ in the second. Other Type III, and all Type IV faults, except the two exceptions mentioned earlier, will in general require the exclusive-OR or biequivalence comparison organ.

For Type I and II faults in (say) AND gates, a reduction of the redundancy ratio below 2:1 is conceivable if there exist one or more input combinations for which $f = 0$ and for which f is not changed to $\underline{1}$ for *any* fault in the network. For example, any direct realization of the function of three variables

$$f = x(y + z) + yz$$

is such that no Type I or II AND-gate fault can ever give rise to an incorrect output for the input combination $x = y = z = 0$, for which $f = 0$. Similarly, no Type I or II OR-gate fault can give rise to an incorrect output for the input combination $x = y = z = 1$, for which $f = 1$. Thus, these particular input combinations can be treated as "don't cares" in the realization of g_0 or g_1, with the hope of circuit simplification. Unfortunately, no such simplification is possible for this example, or for any simple switching function, even assuming unlimited use of AND and OR gates. There may be some functions of 5 or more variables for which the cost of the g-network is slightly less than the cost of the f-network, but these would be exceptional. Consequently, this possibility is given no further attention.

The possibility of combining the realizations of f and g_1 (or f and $\bar{f}$) to achieve a reduction of cost will be considered in Sec. III.

D. DELAYED FAULT DETECTION

The method just described for the detection of either AND- or OR-gate faults, but not both, would find its main direct application in those networks containing a relatively small number of OR- or AND-gates, respectively, which might therefore be individually fault-masked at a small cost. For example, faults in the single k-input OR-gate in a two-level AND/OR network having k AND-gates can be masked with an additional $k + 2$ rectifiers, by the arrangement already introduced in Fig. 1(d). While not all single faults are detected by this arrangement, those not detected will never cause an erroneous output.

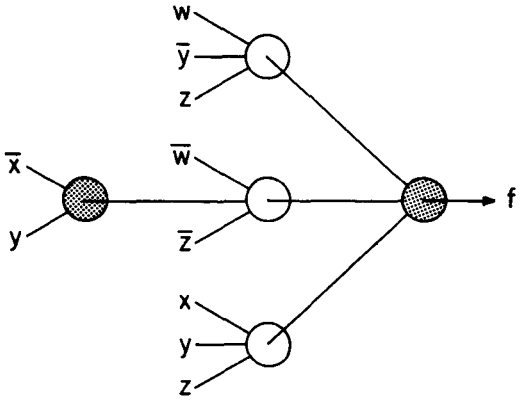


FIG. 5

AN IRREDUNDANT AND-OR REALIZATION OF THE EXAMPLE IN THE TEXT

However, our main interest in faults in AND-gates only or OR-gates only lies in the possible simplifications in the lower network when it is not insisted that *immediate* indication of the fault be provided. We have already observed that a single AND-gate fault will cause a $\underline{0}$ to change to a $\underline{1}$ in the network output, generally for not just a single, but several, input-variable combinations. For cases in which it can be assumed that (1) the input combinations are cycled through all possibilities either intentionally or in the course of normal use of the network, and (2) delayed indication of a fault is satisfactory, then the procedure about to be described may allow a considerable saving in cost of the lower network, depending upon the function f.

A switching function such as

$$f = w\bar{y}z + \bar{w}(\bar{x} + \bar{y})\bar{z} + xyz$$

for which a realization is shown in Fig. 5, may be described by its truth table, or table of combinations

w	x	y	f	f
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
	etc.			
1	1	1	1	1

or, more compactly by its *characteristic number*

$$f = [1010\ 0011\ 0100\ 0101]$$

which is a horizontal display of the f-column of the table of combinations. The successive 2^n digits of this representation of f describe the value of f, the output of the network realization of f, when each of the combinations of the n input variables is applied to the network, in binary counting order.

If a Type I fault occurs at the AND-gate input designated w at the top of the network of Fig. 5, the resulting output function

$$f_{e_1} = [\underline{1}110\ 0\underline{1}11\ 0100\ 0101]$$

is seen to differ from f in that some of its $\underline{0}$'s have been changed to $\underline{1}$'s, as observed earlier, and as indicated here by the underlined digits. Considering the entire set of possible faulty outputs f_{e_1}, f_{e_2}, ... f_{e_9} due to Type I AND-gate faults listed in Table I, we may state the conditions on the formation of g_1 compactly as follows:

(1) Set $g_1 = 0$ for every input combination for which $f = 1$.

(2) For each possible fault, set $g_1 = 1$ for *at least one* input combination for which $f_e = 1$, $f = 0$.

(3) Treating all other input combinations for which $f = 0$ as "don't cares," form an economical realization of g_1.

One possible selection of g_1 is listed in Table I, and is

$$g_1 = w\bar{z} + \bar{x}yz + \bar{w}x\bar{y} \quad .$$

Its realization is shown in Fig. 6, and requires 11 rectifiers, in comparison with the 14 rectifiers required for f (Fig. 5), and the 15 which would be needed for a realization of $\bar{f}$ in similar form.

TABLE I

ITEMIZATION OF ERRONEOUS OUTPUTS DUE TO AND-GATE FAULTS IN THE NETWORK OF FIG. 5

f	=	[1 0 1 0 0 0 1 1 0 1 0 0 0 1 0 1]	
f_{e_1}	=	[1 $\underline{1}$ 1 0 0 $\underline{1}$ 1 1 0 1 0 0 0 1 0 1]	w
f_{e_2}	=	[1 0 1 0 0 0 1 1 0 1 0 1 0 1 0 1]	$\bar{y}$
f_{e_3}	=	[1 0 1 0 0 0 1 1 $\underline{1}$ 1 0 0 $\underline{1}$ 1 0 1]	z
f_{e_4}	=	[1 0 1 0 0 0 1 1 $\underline{1}$ 1 $\underline{1}$ 0 0 1 1 1]	$\bar{w}$
f_{e_5}	=	[1 0 1 0 1 0 1 1 0 1 0 0 0 1 0 1]	$(\bar{x} + y)$
f_{e_6}	=	[1 $\underline{1}$ 1 $\underline{1}$ 0 0 1 1 0 1 0 0 0 1 0 1]	$\bar{z}$
f_{e_7}	=	[1 0 1 $\underline{1}$ 0 0 1 1 0 1 0 $\underline{1}$ 0 1 0 1]	x
f_{e_8}	=	[1 0 1 0 0 $\underline{1}$ 1 1 0 1 0 0 0 1 0 1]	y
f_{e_9}	=	[1 0 1 0 0 0 1 1 0 1 0 0 0 1 $\underline{1}$ 1]	z
g_1	=	[0 x 0 1 1 1 0 0 x 0 x 1 x 0 1 0]	
		at least one of these	
g_1	=	[0 0 0 1 1 1 0 0 1 0 1 1 1 0 1 0]	

prime implicants	a	b	f	a	e	d	c	e	c
	b	c	g	g		e	d	f	

A systematic procedure for forming g_1 will be described in the next section.

Since for every fault there is at least one input combination for which both $f_e = 1$ and $g_1 = 1$, every one of the faults in the f-network is eventually detected. Since $f = 0$ whenever $g_1 = 1$ in the absence of faults, only these faults can give rise to an error output $e = 1$. Type I AND-gate faults in the g-network are also detected, since any such fault always gives rise to at least one input combination for which $g_1 = 1$. If $f = 1$ for the same case, the fault is detected.

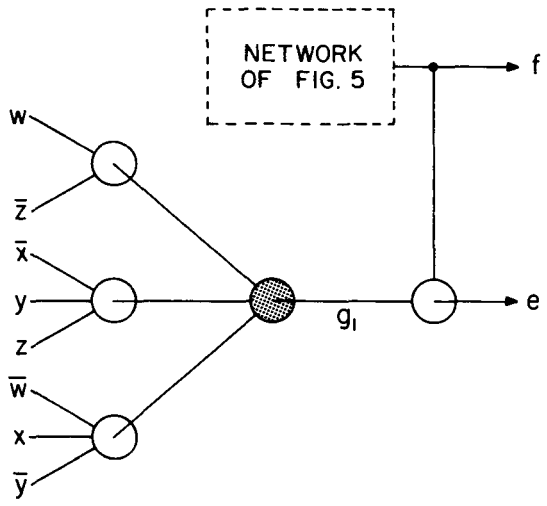


FIG. 6

REDUNDANT NETWORK ARRANGEMENT FOR THE DELAYED DETECTION OF ALL SINGLE TYPE I AND II AND-GATE FAULTS

If $f = 0$ for the same case, the fault is of no consequence and a simpler g-network in which the faulty element is deleted would serve just as well.

The tabular method just described is clearly applicable not only for Type I faults, but also for Type II, and for that matter for any smaller subset of such faults which might be considered as the most likely to occur. To derive g_1, it is only necessary to form the table of erroneous outputs f_{e_1}, f_{e_2}, etc., for this family of faults, instead of for *all* faults as was done in Table I.

Actually the circuit arrangement just derived for detecting Type I AND-gate faults also detects all Type II AND-gate faults automatically. Just as the output function of every fault-free AND-gate implies each of the functions obtained when a Type I or II fault is present, so also does every output function with a Type I fault imply the output function with some Type II fault. The truth of this statement follows from the form of the element output expressions in the AND-gate column of Fig. 3. In an AND-OR network, therefore, protection against Type I faults automatically provides protection against Type II faults.

As described before, the arrangement of Fig. 6 also provides for the detection of Type I and II faults in the comparison AND-gate. Thus, the entire network is protected against all single Type I and II AND-gate faults. The redundancy is 29/16 = 1.81.

If the objective is to detect *only* Type II faults, a simpler g-network is usually possible. This occurs because the number of $0 \rightarrow 1$ changes in f due to a Type II fault is usually much greater than the corresponding number which arise from a Type I fault—a consequence of the fewer number of literals in the gate output function in the former case. Thus, greater freedom is possible in the selection of the input combinations for which $g_1 = 1$, and a simpler g-function is adequate. In fact, if both f and g are realized in the form of two-level AND/OR networks, the AND-gate outputs in the presence of Type II faults become single literals (w, $\bar{y}$, etc.), so that the only conditions which g_1 need satisfy are:

(1) $g_1 = 0$ whenever $f = 1$

(2) Every literal which appears in the expression for f must also appear in the expression for g_1, at least in expanded form.

The first of these conditions is familiar. The sufficiency of the second follows from the recognition that a fault which causes a single literal, such as w, to appear in f will be detected for some input combination, provided only that g_1 contains a term containing w, either explicitly or implicitly..

It is therefore apparent that Type II fault detection generally requires a diminishingly small redundancy ratio as the number n of input variables increases, since g_1 normally need not contain more than $2n$ literals. In fact, if two input combinations can be found among those for which $f = 0$ which have completely complemented literals, the corresponding two products are adequate for the formation of g_1. *E.g.*, for the running example, the function

$$g_1 = \bar{w}\,\bar{x}\,\bar{y}\,\bar{z} + w\,x\,y\,\bar{z}$$

or even

$$g_1 = \bar{w}\,\bar{y}\,z + w\,y\,\bar{z}$$

would be adequate. The proportion of switching functions f which do not contain a pair of complementary input combinations in $\bar{f}$ is exceedingly small. Thus, all Type II AND-gate faults in the two-level AND/OR realization of almost all n-variable switching functions can be detected with no more than $2n + 4$ additional rectifiers.

The exact duals of the procedures described above yield network configurations for the delayed detection of all OR-gate failures in AND-OR networks. The following observations are direct parallels of those for AND-gate failures:

(1) Detection of Type I OR-gate failures automatically provides for detection of Type II OR-gate failures.

(2) For detection of Types I and II OR-gate failures, select g_1 to be $\underline{1}$ for every case for which $f = 0$; for each possible fault, set $g_1 = 0$ for at least one input combination for which $f_e = 0$, $f = 1$, treating all other cases for which $f = 1$ as "don't cares."

(3) For detection of only Type II OR-gate faults in a two-level OR/AND network, select $g_1 = 1$ whenever $f = 0$, and utilize in g_1 every literal which appears in $\underline{f}$, explicitly or implicitly..

We have assumed as before that the error output is provided in complemented form: $\overline{e} = g_2 + f$. All Types I and II faults in this comparison organ are also detected.

An attempt to augment a network N with two checking networks, one g_1-network for detecting AND gate faults, and another g_1 network for detecting OR-gate faults, yields a final network which does indeed detect both AND and OR-gate faults in N, but OR-gate faults in the first g_1-network and AND-gate faults in the second g_1-network are left undetected. Although a fourth network might conceivably be added to detect these remaining faults, and all of its own faults, it is difficult to imagine how this could be done with less than 2:1 redundancy.

Consequently, use of $g_1 = \overline{f}$ or $g_0 = f$ with a $\oplus$ or $\equiv$ comparison organ (network redundancy) is recommended for the simultaneous detection of both AND- and OR-gate faults. Again, however, the alternative of masking one of these two kinds of faults while detecting the other is still available, if the application permits, and would generally be expected to be less costly than duplication of the entire network. In particular, any AND-OR network N may be checked with a two-level AND/OR g_1-network and a two-level OR/AND g_1-network, in which the output gates of the checking networks have been fault-masked. Since the complexity of the checking networks generally increases with n at a rate slower than the complexity of N itself, the over-all redundancy ratio for this arrangement may well be less than 2:1. Moreover, this arrangement is still valid for *any* realization of f, using NOR's NOT's, etc., as well as AND- and OR-gates. In this case the two g_1-functions are still formed on the basis of the *added* and *deleted* 1's in the characteristic numbers of network outputs in the presence of faults, f_{e_1}, f_{e_2}, etc., and the tabular method described earlier can still be used. Additional flexibility and hence an additional potential saving now exists, however, because some faults may cause *both* the addition and deletion of $\underline{1}$'s, and may be detected by appropriately located $\underline{1}$'s in the first g_1 *or* $\underline{0}$'s in the second g_1, not necessarily both.

By the same token, any faults of Types III and IV in N can also be detected, provided only that these faults are included with the others when the table of erroneous output functions is compiled. Type III and IV faults in the checking networks will not necessarily be detected, however, and if considered to be sufficiently likely to occur, must be protected by other means, or through the use of network redundancy as described at the beginning of this section.

To achieve the greatest over-all economy in a redundant circuit, one should properly consider also certain non-minimal realizations of the irredundant portion (f-network), since it is conceivable that a small increase in the cost of realizing f may allow an even greater decrease in the cost of the g-network.

While this possibility must be acknowledged, our experience with numerous examples tends to indicate that it is only rarely the case that such a simplification will occur in circuits of moderate complexity. One of these exceptional examples is provided by the function

$$f = \bar{w}x\bar{z} + xyz + w\bar{y}z + \bar{x}\bar{y}\bar{z}$$

which requires, by the procedure described above, a checking function

$$g_1 = \bar{f} = \bar{w}\bar{y}z + x\bar{y}\bar{z} + wy\bar{z} + \bar{x}yz \quad .$$

Realizations of f and g_1 each require 16 rectifiers. However, the longer expression for the same function

$$f = xyz + w\bar{y}z + w\bar{y}\bar{x} + \bar{w}\bar{x}\bar{z} + \bar{w}y\bar{z}$$

which may also be written

$$f = xyz + w\bar{y}(\bar{x} + z) + \bar{w}\bar{z}(\bar{x} + y)$$

requires a checking function

$$g_1 = wx\bar{z} + \bar{x}yz + \bar{w}x\bar{y} \quad .$$

Here f and g_1 require 16 and 12 rectifiers, respectively.

E. MINIMIZATION OF THE REDUNDANT NETWORK

A systematic procedure will now be described for the derivation of a g_1-function which has a minimal two-level AND/OR realization. This procedure bears a close resemblance to McCluskey's method for the selection of a minimal set of prime implicants equivalent to a given function, from a complete prime implicant table for the function.[3] We will describe the procedure in connection with the running example, for which the error-outputs

due to AND-gate faults are listed in Table I; its generalization to arbitrary functions and different error patterns will be merely indicated, but is direct.

From condition (1) (see p. 14) and the assumed form of the g_1-realization, it follows that the expression for g_1 will take the form of a sum of products (of input variables and their complements) which implies $\bar{f}$. By condition (3), each such product will be a *prime implicant* of $\bar{f}$ (that is, a shortest product which implies $\bar{f}$). Only those prime implicants need be included in the sum which are adequate to satisfy condition (2); that is, at least one of the fundamental products (input combinations) for which $f = 0$ and $f_e = 1$ must imply this sum of prime implicants, for every possible fault being considered.

To arrive at such a sum, list for each input combination for which $f = 0$ all of the prime implicants of $\bar{f}$ which are implied by the corresponding fundamental product. *E.g.*, for input 0001 (the second column of Table I), the prime implicants

$$\bar{w}\,\bar{y}\,z = a$$

and

$$\bar{w}\,\bar{x}\,z = b$$

are implied. The other $f = 0$ columns are similarly indicated, with the notation:

$$c = \bar{x}\,y\,z \qquad e = w\,\bar{z}$$

$$d = w\,\bar{x}\,y \qquad f = w\,\bar{y}\,\bar{z} \qquad g = \bar{w}\,x\,\bar{y} \quad .$$

Following McCluskey, we now form a symbolic logical function which expresses all of the alternative sets of prime implicants which must be included in the sum. Using upper-case letters A, B, ..., to represent binary variables whose values (T and F, or $\underline{1}$ and $\underline{0}$) indicate inclusion or exclusion, respectively, of the prime implicants designated by corresponding lower-case letters a, b, ..., this function may be written

$$L(g_1) = (A_v B_v G)\,(C_v D)\,(E_v F)\,(E_v D)\,(F_v G)\,(A_v B_v C)\,(B_v C_v D)\,(A_v G)E \quad .$$

The first sum of this product expresses the requirement that the g_1-sum must contain one or more of the prime implicants a, b, or g if Fault No. 1 is to be detected. Similarly, the second term requires prime implicants c or d to be included in order that Fault No. 2 be detected, etc. Simplification of this expression to a sum of products, and reordering of those products to place the shortest ones first, yields

$$L(g_1) = EGC_v EGDA_v ECAF_v \ldots.$$

Thus, a sum of prime implicants e, g, and c is adequate for g_1. The circuit corresponding to this particular selection is shown in Fig. 6.

The validity of this approach for any function and for an arbitrary set of AND-type faults (or OR-type, by the dual approach, but not both) should be obvious from the validity of the McCluskey procedure itself. In fact it should be noted that the procedure is almost identical to McCluskey's synthesis of a minimal two-level $\bar{f}$-network, differing only in the last stage in the manner of forming $L(g_1)$, instead of $L(\bar{f})$.

III MULTIPLE AND MULTI-OUTPUT GATE-TYPE NETWORKS

A. NETWORK REDUNDANCY

The bank of m single-output networks shown in Fig. 7 may be checked by the simple expedient of (1) adding a single additional network (shown dotted) with the same inputs, and whose output function f_0 is defined by

$$f_0 = f_1 \oplus f_2 \oplus \cdots \oplus f_m = f_0(x_1, \cdots x_n)$$

and then (2) providing an $(m + 1)$-input parity gate to supply the error signal. This arrangement is based on the recognition of the sequence of output values $f_1, f_2, \cdots f_m$ as a code word, to which a redundant parity digit f_0 may be added for single-error detection.[5] Faults of all types which are confined to a single network (even that producing f_0) are detected by this configuration; thus, all *single* faults except some internetwork short circuits and certain failures in the parity checker are detected.

For most switching functions the cost of the multiple network arrangement will tend to increase directly with the number of networks (outputs), leading to a redundancy ratio of about

$$\frac{m + 1}{m} = 1 + \frac{1}{m}$$

not counting the parity gate. An $(m + 1)$-input parity gate is admittedly a costly portion of the circuit, particularly if it must be realized with AND- and OR-gates only. Most of this complexity is inherent, however, in that *any* checking network which must be sensitive to changes in value of one of the output functions f_1, $f_2, \ldots, f_m$, or f_0, for a wide

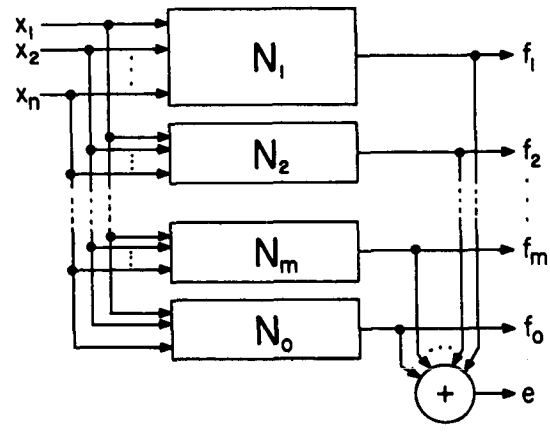


FIG. 7

REDUNDANT ARRANGEMENT FOR THE DETECTION OF A FAULT IN A BANK OF SINGLE-OUTPUT NETWORKS

range of proper values of these outputs, requires a certain minimal extent and type of dependency upon each of the output variables. With this constraint, the parity function is really one of the simplest possible functions, by virtue of its variable symmetries. If one is free to employ certain less common types of logical elements in the checking subnetwork—*e.g.*, threshold elements—then the circuit complexity of the parity gate can be reduced considerably.[6]

If it is desired to detect faults in more than one of the networks, more than a single redundant output may be formed in a similar fashion. In general, all faults in up to q networks may be detected if the redundant outputs are formed in accordance with the structure of a q-error-detecting code—*i.e.*, a code of minimum distance $q + 1$. The reader is referred to the rapidly growing body of literature on coding theory for tables of the number of redundant outputs (check digits) required as functions of q and m, and for the patterns of parity checks which should be used in these cases.* Because of the relative complexity of combinational (parallel) realizations of multiple-error networks and detectors, however, these more sophisticated applications of coding theory for fault detection are economical only for very large networks.

In those parts of a system which consist of a cascade of several banks of networks, one band of which is the type shown in Fig. 7, a simplification is possible by using only a single parity-gate for the entire cascade. In this arrangement the outputs of one bank become the inputs to the next bank in the cascade. Each multiple-network bank will then receive not n, but $n + 1$ inputs, one of which (x_0, say) is a redundant parity variable. The output f_0 of this bank must now be chosen to show a violation of the parity check, not only when a fault in this bank causes one of the outputs $f_1, f_2, \ldots f_m$, or f_0 to be in error, but also when the parity over the input digits $x_1, x_2, \ldots x_n$, and x_0 is violated because of a fault in an earlier bank. Thus, a single fault in any network within any bank of the cascade chain will appear as a violation of the parity check over the outputs of that bank, and this parity violation will be passed from bank to bank to the end of the chain. At this point a single parity gate of the type used in Fig. 7 may be provided to detect the error.

---

* See, for example, Refs. 5 and 7.

The output f_0 will now be the same as before, except for the addition of the parity-check condition on the input variables:

$$f_0 = (f_1 \oplus f_2 \oplus \ldots \oplus f_m) \oplus (x_0 \oplus x_1 \oplus x_2 \oplus \ldots \oplus x_n) \quad .$$

As before, f_0 is to be realized separately, directly as a function of the input variables x_0, x_1, $\ldots$ x_n, and not from the outputs f_1, etc.

In similar fashion, banks of networks arranged in a tree-structure may be checked. If a point of confluence occurs at a bank with (say) two sets of inputs x_0, x_1, $\ldots$ x_n and ξ_0, ξ_1, $\ldots$ ξ_ν, and the usual outputs f_0, f_1, $\ldots$ f_m, then f_0 may be formed as

$$f_0 = (f_1 \oplus \ldots \oplus f_m) \oplus (x_0 \oplus x_1 \oplus \ldots \oplus x_n) \oplus (\xi_0 \oplus \xi_1 \oplus \ldots \oplus \xi_\nu) \quad .$$

Clearly, a single error in one of the f_i, one of the x_i, or one of the ξ_k will be passed on to the last bank for detection.

These same kinds of redundant function may also be used in the case where the bank of single-output networks is replaced by a single, less costly multi-output network, provided that the extent of shared elements in the latter form is not so great that one fault can result in errors at more than a single output. It may be possible to take one of the following steps to alleviate this difficulty:

(1) The multi-output realization may be augmented or modified so that any single fault causes an *odd* number of outputs to be in error. Such an error pattern will, of course, be detected by the parity check.

(2) If delayed detection as described in Sec. II-D above is allowed, the multi-output realization may be arranged so that any single fault causes just one output (or an odd number of outputs) to be in error *for at least one input combination.*

(3) Additional redundant outputs may be provided to allow for multiple error detection, at least for those cases where a single fault causes more than one (or an even number) of outputs to be in error.

As an example consider a code converter which converts decimal numbers in the 8421 binary code into their decimal equivalents in the standard 5-digit teletype code. Here $n = 4$, $m = 5$, and the five output functions are easily determined to be

$$f_1 = y + \bar{w}\bar{x}z$$

$$f_2 = \bar{w}\bar{x}\bar{y} + xyz + \bar{x}\bar{z} + \bar{y}\bar{z}$$

$$f_3 = \bar{w}\bar{x}\bar{y} + xy + w\bar{z}$$

$$f_4 = x\bar{y}\bar{z} + wz$$

$$f_5 = y\bar{z} + \bar{y}z + \bar{w}\bar{x}\bar{y} \quad .$$

The six non-decimal code combinations of the input variables w, x, y, and z have been taken as "don't cares," and these expressions reflect the resultant simplifications which are then possible. (z is the least significant digit.)

For a single bank of independent networks, the redundant output is:

$$f_0 = f_1 \oplus f_2 \oplus f_3 \oplus f_4 \oplus f_5$$

$$= y + xz + \bar{w}\bar{x}\bar{z} \quad .$$

The full, redundant realization in the form of separate two-level AND/OR networks is shown in Fig. 8, and requires 54 gate inputs (*i.e.*, rectifiers). Deletion of the f_0 output saves 8 rectifiers, yielding a redundancy ratio (without the parity gate) of $r = 54/46 = 1.17$. More properly, the redundant network should be compared with the irredundant multi-output version, in which sharing of AND-gate outputs is allowed. The minimum form of this type, shown in Fig. 9, requires 38 rectifiers, and gives the redundancy ratio

$$r = \frac{54}{38} = 1.42 \quad .$$

If this code converter is part of a chain of transducers which are to be checked collectively rather than individually, the inputs w, x, y, and z will be supplemented with a parity digit $x_0 = v$. The redundant output is now given by the longer expression

$$f_0 = (f_1 \oplus f_2 \oplus f_3 \oplus f_4 \oplus f_5) \oplus (v \oplus w \oplus x \oplus y \oplus z)$$

$$= v \oplus [\bar{x}yz + \bar{y}(\bar{w} + x + \bar{z}) + \bar{w}x\bar{z}] \quad .$$

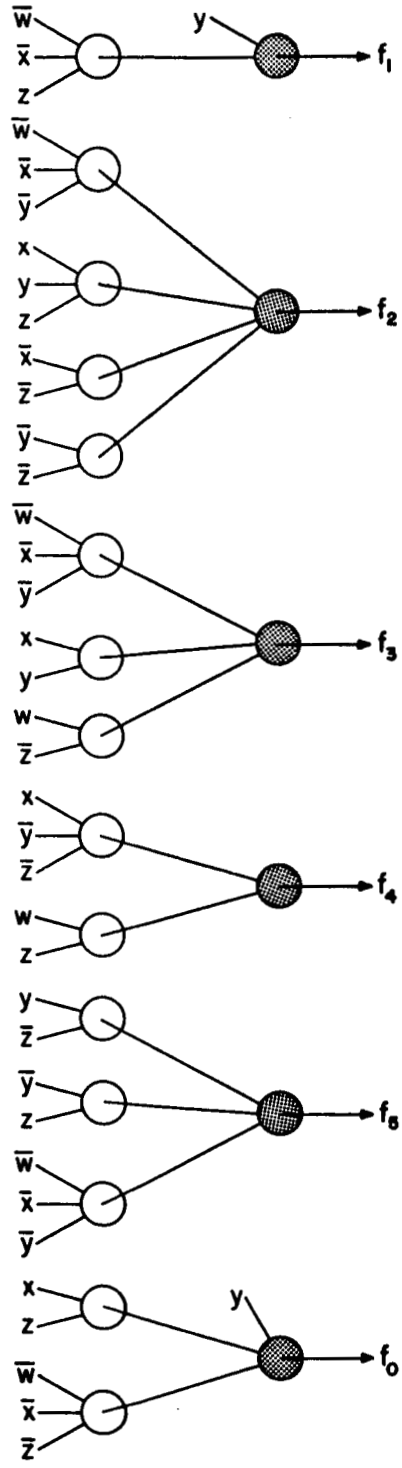


FIG. 8

REDUNDANT AND-OR REALIZATION
OF A CODE CONVERTER

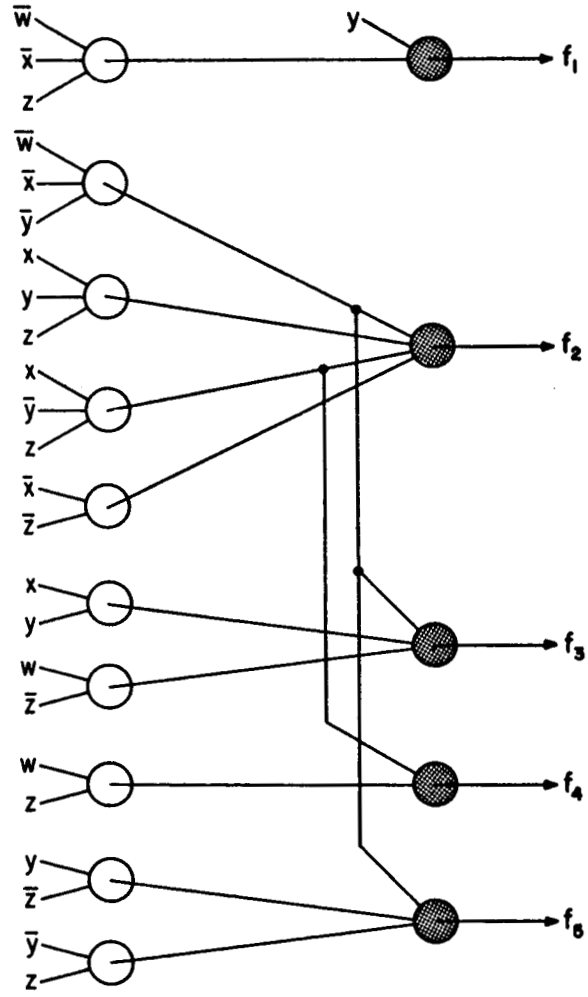


FIG. 9

IRREDUNDANT MULTI-OUTPUT
AND-OR REALIZATION
OF THE CODE CONVERTER

Note that the six "don't care" input combinations cannot now be used to simplify f_0, since their occurrence may accompany a prior fault. A two-level AND/OR realization of f_0 now requires 43 rectifiers, yielding a redundancy ratio

$$r = \frac{46 + 43}{38} = 2.34 \quad .$$

B. ITERATIVE NETWORKS

Iterative networks constitute an important special class of cascaded multiple or multi-output networks, in which each network or bank in the cascade is identical. Iterative networks justify special consideration because (1) they enjoy widespread use as parallel adders, comparators, checkers, and counters, and (2) they can be analyzed by elementary sequential-circuit methods, and therefore serve as an approach to the extension of redundancy techniques to sequential networks.

From this sequential viewpoint, the set of variables $x_1, x_2, \ldots x_n$ may be thought of as a "state," and the cell outputs $f_1, f_2 \ldots f_n$ as a "next state." McCluskey has shown how the behavior of iterative networks may be represented in a *state diagram* of just the type used in the analysis of sequential digital networks.[8]

A redundant state variable x_0 may be added just as previously described, but now with $m = n$, since successive stages of the cascade structure are identical:

$$f_0 = (f_1 \oplus f_2 \oplus \ldots \oplus f_n) \oplus (x_0 \oplus x_1 \oplus x_2 \oplus \ldots \oplus x_n) \quad .$$

The above comments (1) - (3) regarding the sharing of outputs in multi-output realizations continue to hold for the iterative case, and in fact are now somewhat more pertinent in view of the greater ease with which the effects of faults may be determined.

In general, the successive banks or "cells" of an iterative network receive inputs not only from the previous cell, but also by direct external connection. Such an iterative network is shown in Fig. 10, in which these inputs are designated $y_1, y_2, \ldots y_p$.

By way of example, Fig. 11 shows the irredundant and (underneath) the redundant portions of one cell of a particular iterative network. This

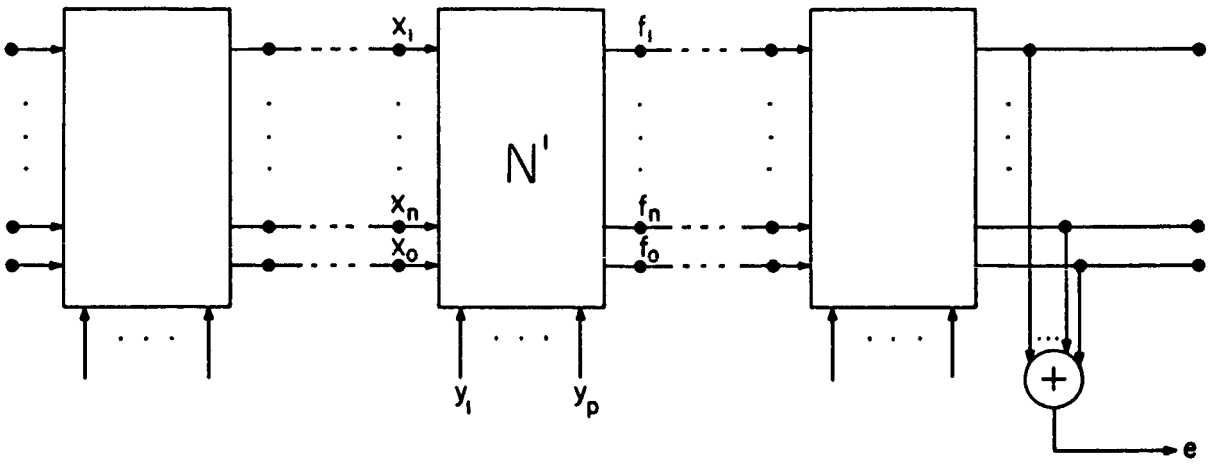


FIG. 10

A REDUNDANT ITERATIVE NETWORK WITHOUT INDIVIDUAL CELL OUTPUTS

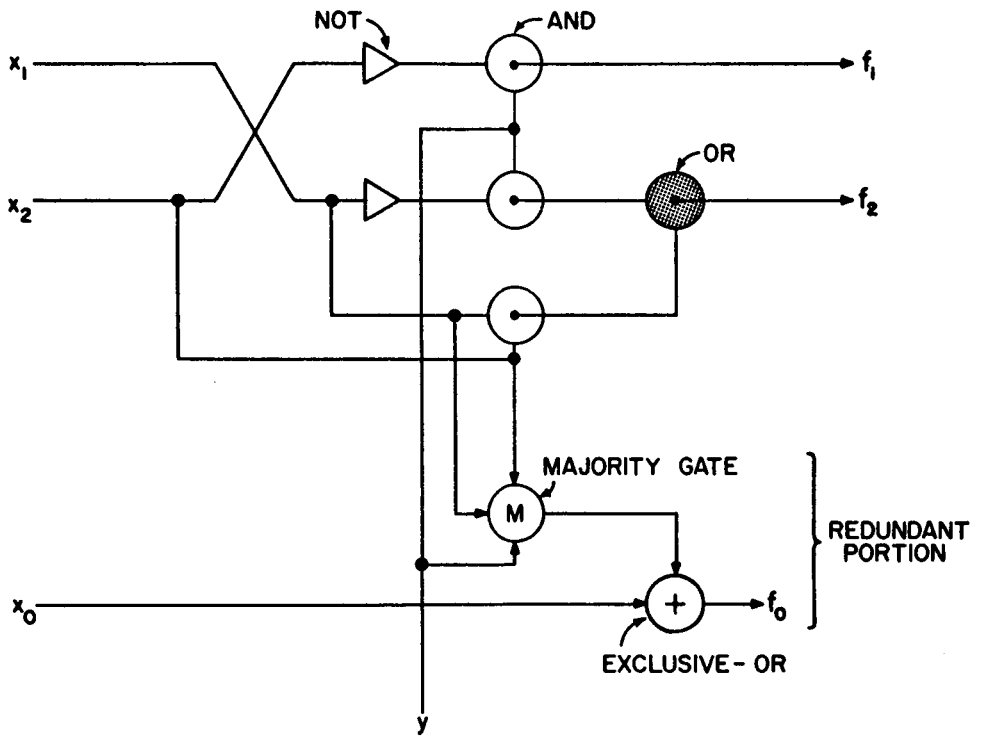


FIG. 11

EXAMPLE OF ONE CELL OF A REDUNDANT ITERATIVE NETWORK WHICH DETECTS THREE ADJACENT 1'S IN THE SUCCESSION OF Y-VALUES

network provides an output $x_2\bar{x}_1$ from the final cell when, and only when, the succession of y-values at cells 1, 2, ... N contains within it at least three 1's in succession. (Simplifications are usually possible in the first and last cells, but these are not specifically shown here.) The state diagram is shown in Fig. 12, and reveals that the state sequence 00, 11, 01, and 10 can occur only when $y = 1$ for three successive cells; any deviation from this pattern causes return to state 00. (The first and second digits of the state number are x_2 and x_1, respectively.) For this particular assignment of binary state numbers to the four states, the next-state equations are easily derived:

$$f_1 = y\bar{x}_2$$

$$f_2 = y\bar{x}_1 + x_2x_1 \quad .$$

The redundant output function is now

$$f_0 = f_1 \oplus f_2 \oplus (x_0 \oplus x_1 \oplus x_2)$$
$$= x_0 \oplus M(x_1, x_2, y)$$

where M designates the majority function, equal to $\underline{1}$ when 2 or 3 out of its three inputs have the value $\underline{1}$:

$$M(x_1, x_2, y) = x_1x_2 + x_2y + yx_1 \quad .$$

A three-input exclusive-OR gate attached to the last cell of the cascade chain will now provide an output e equal to $\underline{1}$ when and only when a single fault occurs anywhere within the iterative network.

When the cell logic contains a part which is independent of the inputs x_1, x_2, ... x_n, depending only on y_1, ... y_p, then a simplification is sometimes possible. An example of such a case is provided by a size comparator, which is essentially a subtractor with the "difference" output deleted—that is, only the carry ("borrow") signal

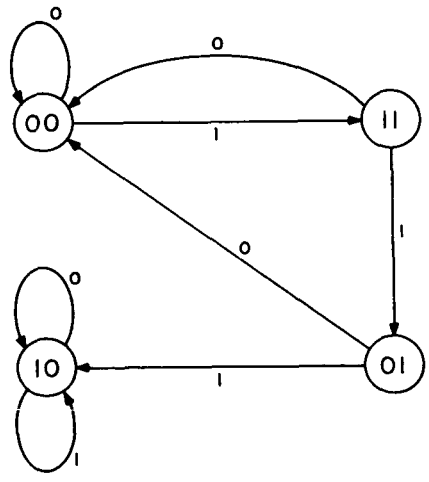


FIG. 12

STATE DIAGRAM FOR THE ITERATIVE NETWORK OF FIG. 11

need be propagated from cell to cell. The equation for the irredundant cell is therefore

$$f_1 = y_1\bar{y}_2 + \bar{y}_2 x_1 + x_1 y_1 = M(y_1, \bar{y}_2, x_1)$$

where y_1 and y_2 are the corresponding digits of the binary numbers being compared. The redundant output of the cell may now be defined by the equation

$$f_0 = \bar{y}_1 y_2 + y_2 x_0 + x_0 \bar{y}_1 = M(\bar{y}_1, y_2, x_0) \quad .$$

This circuit, shown in Fig. 13, is simpler than the redundant cell dictated by the f_1 equation above, which requires the more complex logic for the redundant portion,

$$f_0 = x_0 \oplus x_1 \oplus f_1$$

$$= x_0 \oplus [\bar{x}_1 y_1 \bar{y}_2 + x_1 \bar{y}_1 y_2] \quad .$$

The contribution $y_1\bar{y}_2$ to f_1 is clearly independent of x_1, so that an error in x_1 need not be passed on to the cell output f_1, whenever:

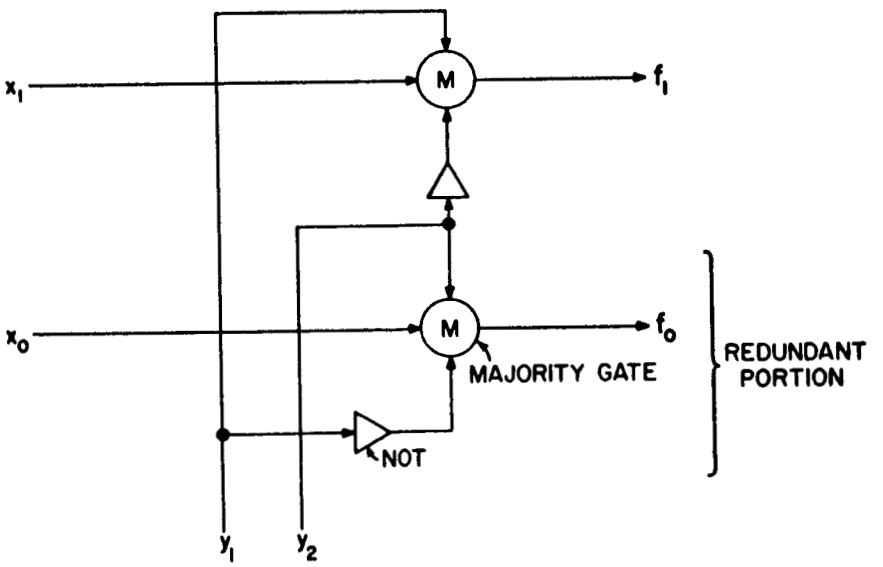


FIG. 13

EXAMPLE OF ONE CELL OF A REDUNDANT SIZE COMPARATOR

(1) $y_1 = 1$, $y_2 = 0$, since $f_1 = 1$ regardless of the value of x_1, or

(2) $y_1 = 0$, $y_2 = 1$, since $f_1 = 0$, again regardless of the value of x_1.

By symmetry of the f_1 and f_0 expressions, the same independence holds for the redundant cell output f_0 with respect to x_0. The final comparator should take the form of a biequivalence gate, since f_1 and f_0 are normally complementary.

In general, iterative networks may provide individual cell outputs, $z_1, z_2, \ldots z_q$, say, as well as network outputs from the last cell. For checking purposes, these z-outputs may be treated merely as additional outputs from the multi-output cell, and may therefore be checked separately from the f-outputs, or provided redundantly without local checking if this is permitted. In either case, we may form the new cell output

$$z_0 = z_1 \oplus z_2 \oplus \ldots \oplus z_q$$

as a function of the x-inputs and y-inputs. If in addition internal checking is desired, it may be incorporated into the cascade check by forming f_0 according to the expression:

$$f_0 = (f_1 \oplus \ldots \oplus f_n) \oplus (x_0 \oplus x_1 \oplus \ldots \oplus x_n) \oplus (z_0 \oplus z_1 \oplus \ldots \oplus z_q) ,$$

in which the f_i are expressed in terms of the x-variables, but the z_j are not.

For example, for a parallel adder, $n = q = 1$ and $p = 2$. Digits y_1, y_2, and z_1 may be identified with the addend, augend, and sum in a single digit position, and x_1 and f_1 are the carry input and output, respectively. The equations for an irredundant cell are well known to be

$$z_1 = y_1 \oplus y_2 \oplus x_1 = \text{sum}$$

$$f_1 = M(y_1, y_2, x_1) = \text{carry} .$$

A cascade of m such cells then constitutes an m-digit parallel adder, as shown in Fig. 14(a). The x_1-input to the first cell is the "add-one" input to the adder (used principally on subtraction) and the f_1 output from the last cell is the overflow indication.

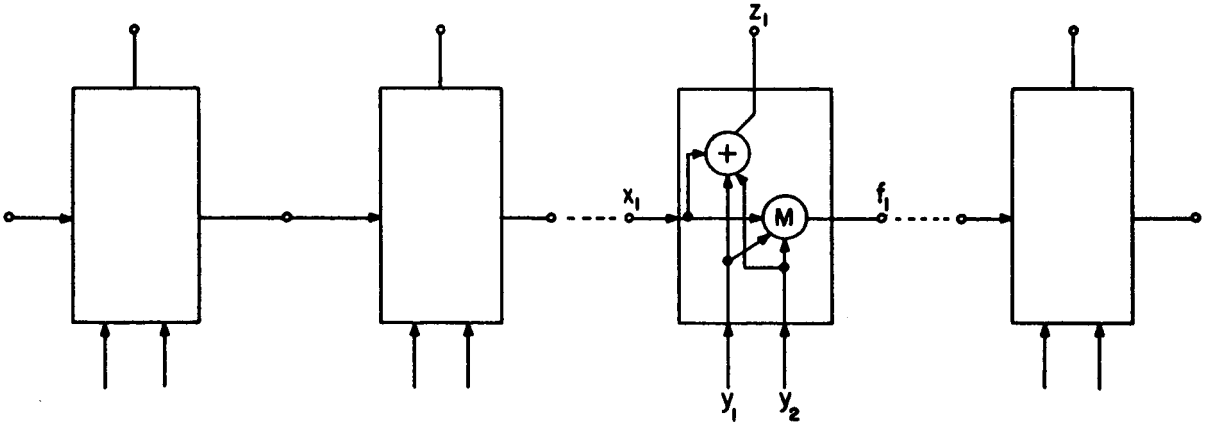


FIG. 14(a)

AN IRREDUNDANT PARALLEL ADDER

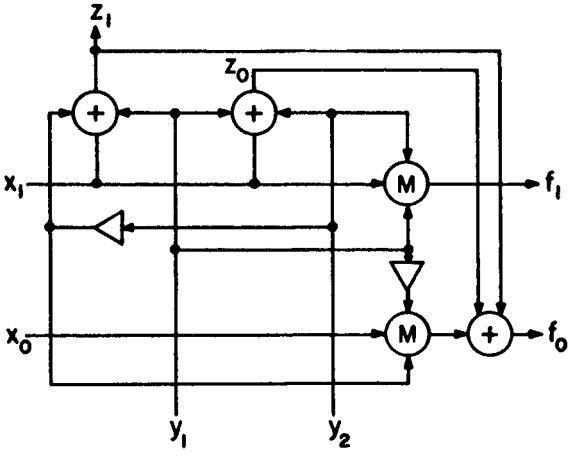


FIG. 14(b)

ONE CELL OF A REDUNDANT PARALLEL ADDER

By the expressions given above, the external redundant cell output z_0 may be selected to be equal to either z_1 or $\overline{z}_1$—*e.g.*,

$$z_0 = \overline{z}_1 = y_1 \oplus \overline{y}_2 \oplus x_1 \quad .$$

The redundant output may be expressed (by analogy with the subtractor just described) as

$$f_0 = M(\overline{y}_1, \overline{y}_2, x_0)$$

or, for internal checking of the z-outputs,

$$f_0 = M(\overline{y}_1, \overline{y}_2, x_0) \oplus z_0 \oplus z_1 \quad .$$

A complete cell based on this latter form is shown in Fig. 14(b). A 2-input $\oplus$-gate at the output of the last cell will be adequate for checking.

C. COMPLEMENTARY-OUTPUT NETWORKS

Certain types of multi-output networks lend themselves to particularly simple redundant configurations, because of already existing redundancy in the output functions. One such class in the family of *complementary-output* networks: those having two outputs, $f_1 = f$ and $f_2 = \overline{f}$. So long as the two outputs are formed independently, the arrangement introduced earlier [Fig. 2(a)] with $g_1 = f_2 = \overline{f}$ applies, and a simple biequivalence gate

$$e = (f_1 \equiv f_2)$$

is adequate to detect all single faults except internetwork short circuits. If the two outputs share a portion of the total circuitry, simple comparison will not always be adequate, since a fault in the shared part may change *both* outputs for at least one input combination, and will go undetected. In the extreme case in which f_2 is formed from f_1 with a single NOT element, the biequivalence gate would detect only one fault: a defect in the NOT element itself.

With regard to the possibility of sharing, it is not difficult to see that no fault in a purely AND-OR network can ever cause one output to change from $\underline{0}$ to $\underline{1}$ and the other to change from $\underline{1}$ to $\underline{0}$ at the same time. This result depends on the already noted "positive" dependence of the

network outputs on the gate suboutputs. If the AND-OR network has only two levels, this property is of no avail, since sharing is impossible on purely logical grounds: no 1st-level gate output can contribute to both f_1 and f_2 in the output level, since it would contribute to them equally (both $\underline{1}$ in an AND/OR network, or both $\underline{0}$ in an OR/AND network). An example of a minimal multilevel AND-OR network with shared gatery is shown in Fig. 15. The comparator attached to the output detects all single faults, except possibly certain short circuits between different portions of the network.

Consider next the circuit arrangement shown in Fig. 16 in which a shared subnetwork N_0 provides outputs h_0 and $\bar{h}_0$, which are in turn used

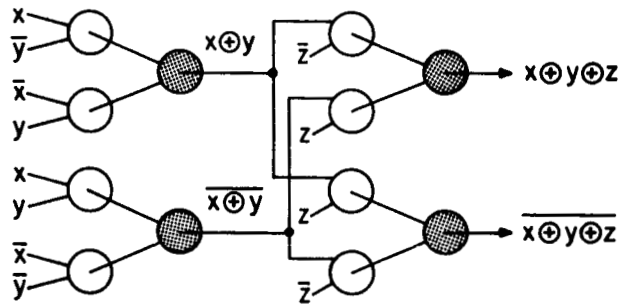


FIG. 15

EXAMPLE OF THE REALIZATION OF A COMPLEMENTARY-OUTPUT NETWORK WITH SHARED ELEMENTS

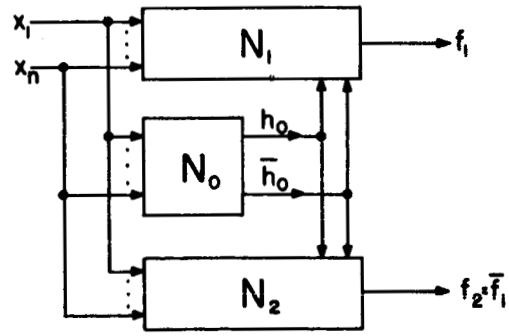


FIG. 16

FORM OF A COMPLEMENTARY-OUTPUT NETWORK HAVING A SHARED SUB-NETWORK

by subnetworks N_1 and N_2 to produce outputs f_1 and $f_2 = \overline{f}_1$, respectively. We no longer restrict ourselves to purely AND-OR networks. Regardless of the form of the upper and lower networks, the dependence of f_1 upon h_0 may be expressed with complete generality as

$$f_1 = \overline{h}_0 h_1 + h_0 h_2$$

and the corresponding expression for $f_2 = \overline{f}_1$ must then take the form

$$f_2 = \overline{h}_0 \overline{h}_1 + h_0 \overline{h}_2 \quad .$$

Here h_1 and h_2, like h_0, are functions of the input variables. For input combinations which cause h_1 and h_2 to be the same (both $\underline{0}$ or both $\underline{1}$), f_1 and f_2 are independent of h_0, and the shared portion is not used. For input combinations which cause h_1 and h_2 to be different, we have either $f_1 = h_0$, $f_2 = \overline{h}_0$, or $f_1 = \overline{h}_0$, $f_2 = h_0$. In either case, simultaneous errors in f_1 and f_2 will occur when and only when a fault causes simultaneous errors in h_0 and $\overline{h}_0$. Thus, the possibility of this undesirable error pattern in *any* complementary-output subnetwork N_0 is reflected in the entire network, and N_0 must be realized so that simultaneous errors in h_0 and $\overline{h}_0$ cannot occur. The shared arrangement of Fig. 16 may still be the most economical, however, as the example already presented in Fig. 15 shows. In fact, complementary-output networks could also be used for forming the functions h_1 and $\overline{h}_1$, and for h_2 and $\overline{h}_2$, and it would be natural to do so if either or both of these pairs of functions contain a common, potentially shared part. These latter networks need not be of the "no simultaneous error" type, however.

Finally, suppose that the $\overline{h}_0$ output of N_0 were omitted in Fig. 16, so that only the h_0-portion is shared. In general, the dependence of f_1 and f_2 upon h_0 is still given by the equations cited above. We distinguish two cases:

(1) Either or both of the networks N_1 and N_2 form their $\overline{h}_0$ term from h_0 with a simple NOT element. For some input combinations, a failure in N_0 will then change the value of both f_1 and f_2 if it has any effect at all, and will go undetected. Thus, this arrangement is not allowed.

(2) Both of the networks N_1 and N_2 realize the $\overline{h}_0$ term, either separately within N_1 and N_2, or combined with

h_1 and $\overline{h}_1$, respectively. Thus, an error in h_0 cannot affect f_1 and f_2 oppositely, so this shared arrangement is always satisfactory.

In summary, then, for a complementary-output network which is to have the "no simultaneous error" property, a shared subnetwork N_0 is allowable provided (1) N_0 is itself a complementary-output network with the same property, or (2) N_0 has a single output, but this output is not complemented directly or indirectly in the balance of the network. In the class of AND-OR networks, shared networks are allowable under all conditions, but will never be needed unless the number of gate levels exceeds two.

D. COMPLETE DECODING AND RELATED NETWORKS

Another important class of multi-output networks is the *complete decoding network*, which consists of an array of AND-gates arranged to provide a 1 on a separate output for each of the 2^n possible combinations of n input variables. Optimal network forms are known for several cases; these minimize the total number of AND-gates, the total number of 2-input AND-gates, or the total number of AND-gate inputs. This class of networks appears in practice in memory access switches, certain code converters, and instruction decoders in digital computers.

The circuit shown in Fig. 17 may be used to detect faults in a complete decoding network, or for that matter in any multi-output network for which one and only one output has the value 1 for each possible input

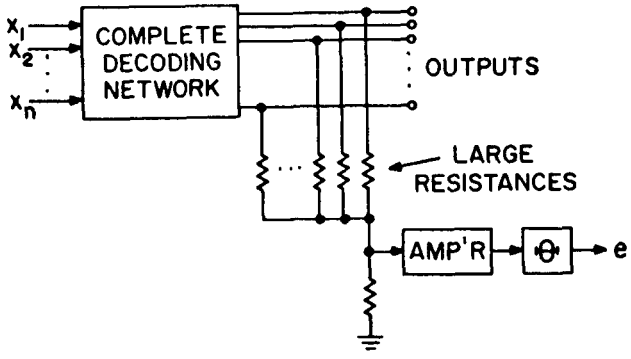


FIG. 17

FAULT-DETECTING CONFIGURATION FOR A COMPLETE DECODING NETWORK

combination. (The complementary-output network is an extreme example of this type of network.) The resistances shown form a resistive adder, whose output s may be amplified (if necessary) and applied to a threshold device⊖. If all faults are of the type which cause additional $\underline{1}$-outputs, such as Type I and II faults in AND-gates, then⊖need only check for an increase in s above the value corresponding to a single $\underline{1}$-output. If a fault can cause either an increase or decrease in the number of $\underline{1}$-outputs, but not both, then⊖must be able to sense any change in the value of $\underline{s}$. For this, a simple bridge circuit is adequate. If certain simple faults can result in a shift of output (that is, a change from $\underline{1}$ to $\underline{0}$ in one output simultaneous with a change from $\underline{0}$ to $\underline{1}$ in another), then the network cannot be fault-checked merely by tests applied at its output terminals, and the circuit shown is inadequate. However, this type of error cannot arise from a single fault in the AND-gate type of complete decoding tree, or in any multi-output network which is made up entirely of AND and OR gates. If NOT or other logical elements are also used, then either these outputs must be formed individually, or some checking scheme which requires the introduction of internal redundancy or redundant outputs must be employed.

IV BRANCH-TYPE NETWORKS

Branch-type combinational switching networks are typically composed of relay contacts, mechanical switches, transistors connected in the so-called "direct coupled" manner, or cryotrons. The logical actions of AND and OR are achieved through series and parallel connections of the basic switching elements, each of which is controlled by one of the input variables or its complement. Other logical operations must be obtained by composition of AND's and OR's, or by allowing sequential behavior, or occasionally through the use of synthetic variants of the original physical elements (*e.g.*, cryotrons with double control windings). These variants are not usually considered to be very practical, and we will not assume their existence in this discussion.

The types of element faults of concern in branch-type circuits are open circuits ("opens"), short circuits ("shorts"), and interelement short circuits. As with gate-type networks, the latter will be assumed here to be much less likely to occur than the former two, and will not be the subject of particular attention.

We first observe that the effect of an open in a branch-type network cannot result in the completion of transmission paths which were not present without the open. Hence, any open which affects the transmission function at all must cause some $\underline{1}$'s to change to 0's, but no $\underline{0}$'s to $\underline{1}$'s, in the characteristic number of the function. Similarly any short will cause some $\underline{0}$'s to change to $\underline{1}$'s, but no $\underline{1}$'s to $\underline{0}$'s. Thus *opens* and *shorts* correspond to (Type I and II) failures in OR and AND gates, respectively, in AND-OR gate-type networks. Consequently, subject to finding a suitable over-all redundant configuration, including the comparator, for branch-type networks, we may translate the pertinent results for AND-OR gate-type networks directly to branch-type networks.

While the g_1-network presents no particular problem, the realization of the comparator normally requires a departure from pure branch-type structures. For the detection of opens only, the error output

$$e_2 = \overline{f}\,\overline{g}_1$$

demands that $\overline{f}$ be directly derivable from f, which is not possible in a pure branch-type structure. Alternatively, the complement of the error output

$$\overline{e}_2 = f + g_1$$

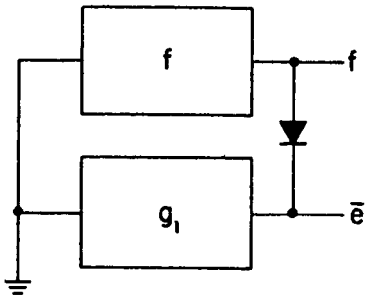
requires a paralleling of the f and g_1 networks, an arrangement which would make the f-output unavailable. The only way to avoid this difficulty is to revert to an electrical "trick," or some form of gate-type structure. If signal conditions permit, a rectifier may be inserted between the f and g_1 terminals [Fig. 18(a)]. Alternatively, an electrically compatible OR-gate may be employed, as shown in Fig. 18(b). If error outputs from several networks are to be combined together, the rectifier approach leads to the series arrangement of Fig. 18(c), while the latter approach using an OR-gate requires only that the set of individual error outputs be combined together in an AND-gate.

For the detection of shorts only, for which the error is given by the expression

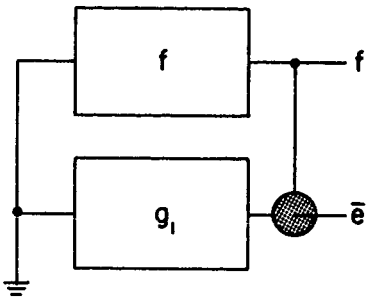
$$e_1 = f g_1$$

the circuit configuration of Fig. 19(a) is adequate. Alternatively, an AND-gate could be used [Fig. 19(b)]. If several error outputs are to be combined, they may be connected directly as in Fig. 19(c), if desired, or combined in an OR-gate. In the former case, sneak paths may exist between the outputs, but only when (1) all outputs so involved have the value $\underline{0}$, or (2) when $e = 1$. Thus, the network is not disjunctive, but as it stands it will always give the correct output when no faults are present.

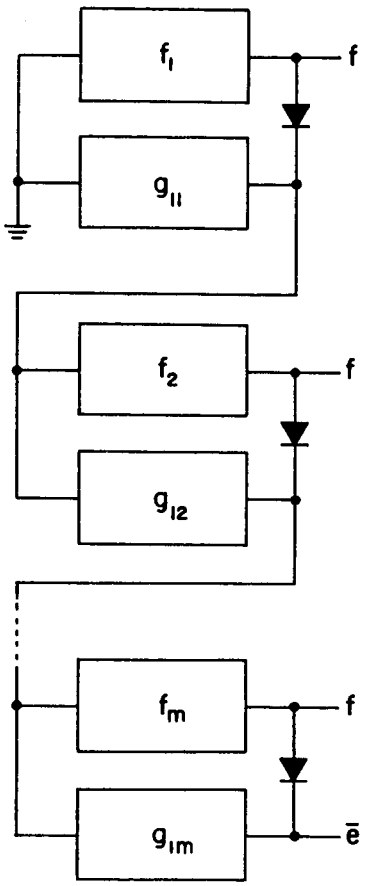
For the detection of both opens and shorts, the parallel and series methods cannot be used separately, since shorts in one checking network and opens in the other will go undetected. The only possibility for a non-gate-type comparator seems to be a bridge circuit with a high-impedance central arm (E), operating on the outputs f and $g_0 = f$, such as is shown in Fig. 20. Clearly, any fault in either the f or the g_0 network will unbalance the bridge, and indicate an error. Even here, combination of several error outputs requires conversion to gate-type signals.



(a)



(b)



(c)

FIG. 18

NETWORK CONFIGURATIONS FOR THE DETECTION OF OPENS ONLY IN BRANCH-TYPE NETWORKS

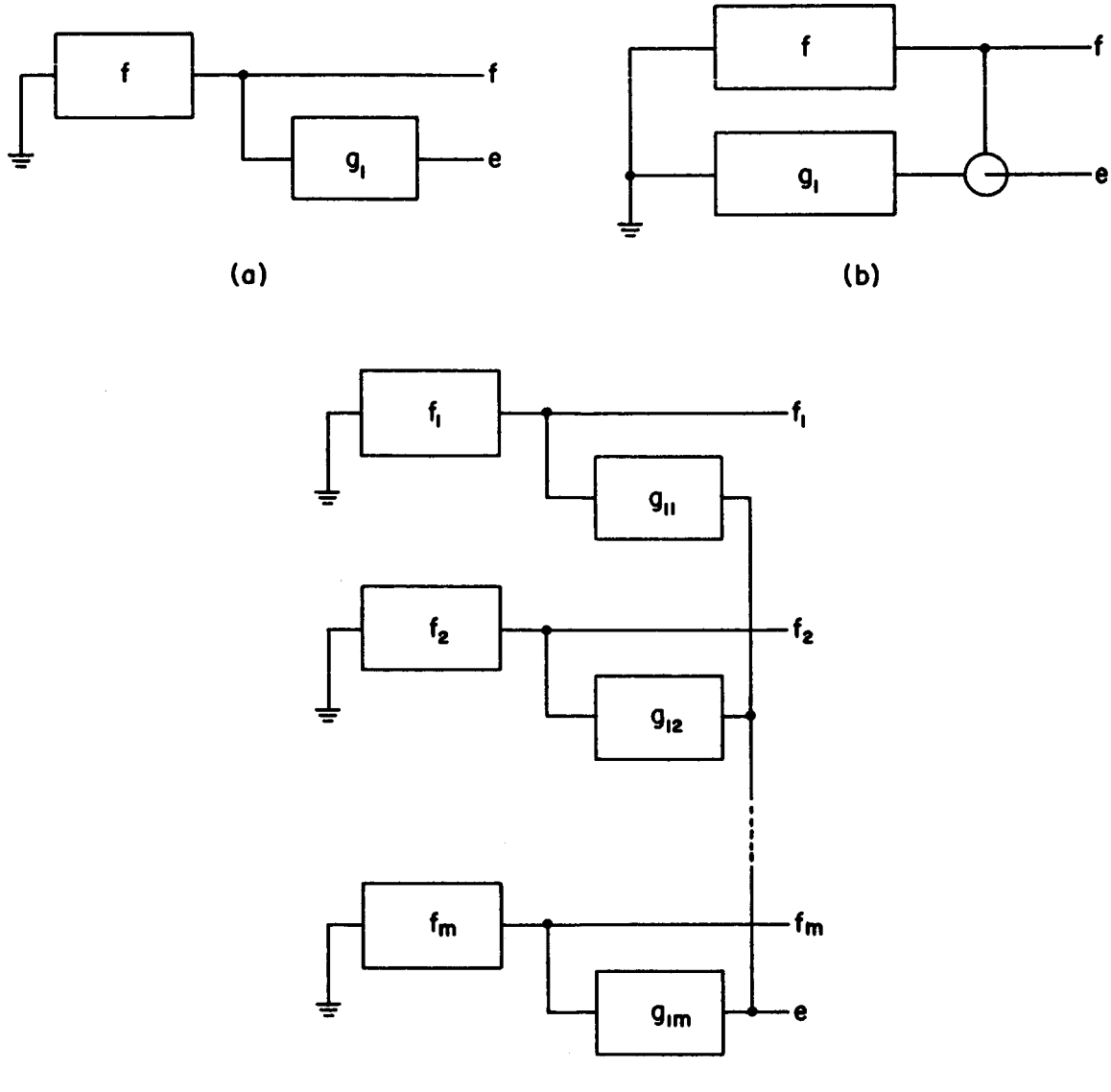


FIG. 19

NETWORK CONFIGURATIONS FOR THE DETECTION OF SHORTS ONLY IN BRANCH-TYPE NETWORKS

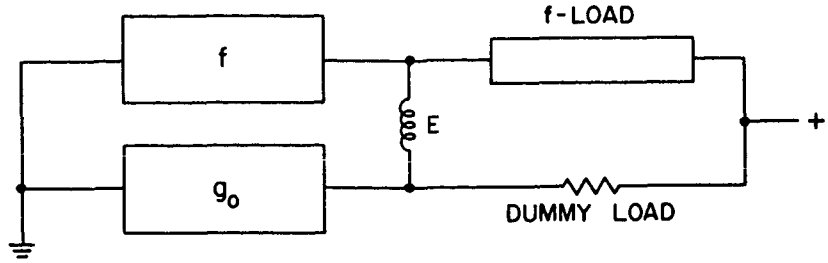


FIG. 20

ONE POSSIBLE CONFIGURATION FOR DETECTION OF BOTH OPENS AND SHORTS IN BRANCH-TYPE NETWORKS

Results on delayed detection for gate-type networks continue to hold without change for branch-type structures, except that the procedure described in Sec. II-E for the formation of a minimal 2-level checking network will not now necessarily lead to a minimal branch-type circuit. In the absence of any available systematic synthesis methods for minimal branch-type networks, however, the procedure should be of some help in deriving an economical realization.

For multiple networks operating from the same input variables, and for multi-output networks containing a sufficient degree of independence of output formation, the use of an auxiliary f_0-output is still valid and economical, but the comparator which parity checks the entire set of outputs must almost certainly be a gate-type element.

Multi-output networks for which one and only one output has the value 1 for each input (such as contact trees) can be checked with a resistive adder and a threshold element, as described earlier. Also, as observed for complementary-output (f-$\overline{f}$) networks of the AND-OR type, unlimited sharing is allowed, since a single fault can never cause both outputs to change at the same time. Such sharing can often reduce considerably the total number of elements in comparison with separate realizations of f and $\overline{f}$ (as proposed above), thereby keeping the redundancy ratio below 2:1 for the detection of all single faults—opens and shorts. The comparator either must be of gate-type, or must be realized in the form of the previously mentioned bridge network.

For example, the parity function

$$f = x \oplus y \oplus z$$

whose gate-type realization in a shared $f - \bar{f}$ network was shown in Fig. 15, is realized with shared branch-type elements in Fig. 21. This configuration requiring 10 elements is minimal, and the minimal single-output network requires 8 elements. Viewed as a single-output network to which network redundancy ($g_1 = \bar{f}$) has been applied, this circuit gives a redundancy ratio (without the comparator) of $r = 10/8 = 1.25$. For an n variable parity function,

$$r = \frac{4n - 2}{4n - 4}$$

$$= 1 + \frac{1}{2(n - 1)} \quad .$$

In iterative branch-type networks, the bilateral nature of the branch-type element limits so severely the class of logical operations which can be performed in a cell that one must almost always resort to the use of a number n of cell-input and -output digits equal to the total number of states.[8] Such a "one-out-of-n" coding of the states is inherently very redundant, however, so that any single fault is easily detected, without extra digits as follows:

(1) Any open-circuited element which has any erroneous effect at all will reduce the number of 1-digits at the cell output from one to zero, and this effect will be propagated to the final cell of the iterative network.

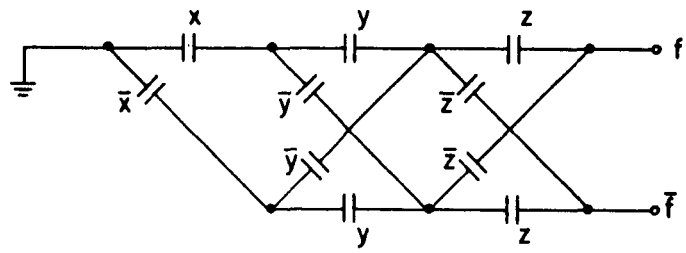


FIG. 21

EXAMPLE OF A BRANCH-TYPE REALIZATION OF A COMPLEMENTARY-OUTPUT NETWORK WITH SHARED ELEMENTS

(2) Any short-circuited element, or any inter-element short circuit, which has any erroneous effect at all will increase the number of 1-digits at the cell output above one, and this effect will be propagated to the final cell.

Therefore, the detector at the output of the final cell need only check on the number of digits $f_1, f_2, \ldots, f_n$, which have the value 1. If this number is not unity, a fault has occurred, and all single faults will cause such a violation of the check if they effect any network output signal.

This result actually assumes a few restrictions on network form, which are almost always satisfied in practice, or may be satisfied at a slight additional cost in the complexity of the cell.[8] In partial compensation, however, we can show that if only a single output from the final cell is needed, then all other final-cell outputs may be connected together. Thus, the end-cells of the network may be simplified, and a single 2-input biequivalence (≡) gate is adequate for fault detection in this case.

Fig. 22 displays one cell of an iterative contact network realization of the "three 1's in a row" example introduced earlier (Fig. 11) for gate-type structures. The biequivalence gate appears at the output of the last cell. The usual end-cell simplifications are not specifically shown.

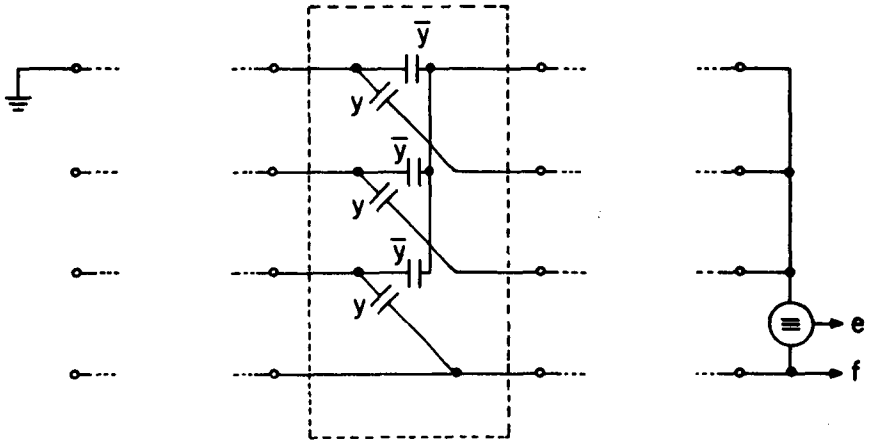


FIG. 22

AN ITERATIVE NETWORK REALIZATION USING BRANCH-TYPE ELEMENTS

V DISCUSSION

We have presented a number of techniques for detecting any single fault in a combinational switching network, with emphasis on reducing the redundancy ratio below the known upper bounds. For single-output networks, the greatest reduction occurs when delayed detection of the fault is permitted. In multiple and multi-output networks, most of the reduction can be achieved by checking the entire set of outputs simultaneously. This is particularly the case when the output function set contains some inherent redundancy, as in complementary-output, decoding, and related networks. In iterative and certain other similar network structures, a cascade of such banks may be checked all at once. Smaller reductions can often be realized by taking advantage of inherent redundancy in the structure of the irredundant network, particularly in branch-type networks. Checking only a limited class of faults can yield a varying reduction in the redundancy ratio, depending upon the switching functions being realized, the type of faults considered likely to occur, and the level on which the redundancy is applied.

Most of the discussion of the preceding sections has centered around redundancy techniques for fault detection on the level of the network or sub-network, rather than the individual logic element, or larger portions of the digital system. While gates and circuit components can be adequately fault-masked with a redundancy ratio of from 2:1 to 4:1 (*e.g.*, as in Fig. 1), the detection of faults on this level is electrically very difficult, if not impossible, because of the need of deriving and combining error signals from each individual gate or component to be checked. Consequently, fault-detection should be applied at the level of the network or higher.

It has already been observed that the cost of detecting the first (if increasingly more likely) fault in a network becomes fractionally less as the size of the network is increased. This is in analogy with the use of a simple parity digit to detect a single error in an indefinitely long code word. So long as the probability of simultaneous double, triple, etc., errors is truly negligible, one should apply fault-detecting redundancy at as high a level as possible, limited only by the cost of the

checking network and the cost of the redundant portion of the network. This latter limit may well set the optimum level below that of the entire digital system, depending on the degree of data reduction as opposed to dispersion of signals within the system. Thus, a system which performs a large amount of signal analysis to either reduce or compact in the output the informational content of the input data will generally allow the smallest redundancy ratio when most of the system is checked at once. However, a highly synthetic and distributed machine such as a general-purpose digital computer is probably best checked by local fault detectors operating separately over memories, control organs, input and output data transfers, etc.

Because of the great variety in size, function, structure, and mode of operation of digital systems in general, such statements as these must be highly qualified if they are to be rigorously applicable in general. Pending a deeper analysis of fault-detecting redundancy in digital systems as a whole, we might best accept these observations for the time being merely as suggestions and "rules of thumb" rather than as a rigid procedure. Also to be considered in this vein are the following well-known or rather obvious possibilities for applying such redundancy on the system level:

(1) Error-detecting codes can be applied to check simple transfers of data between registers, memories, input, output, and recording media.[7] Serial, parallel, and hybrid versions of these codes should be considered, as well as both binary and non-binary (*e.g.*, modulo-nine) interpretations. The only present precaution which must be taken in this more standard use of codes is the detection of faults in the error-detecting circuitry itself. Here, simple duplication or possibly an application of one of the methods described in Sec. II may be used.

(2) A loop check will often allow for the economical detection of almost all faults in a subsystem or network whose inverse operation is more easily realized than the operation itself. Typical of such operations are division, solution of linear equations, address decoding, and analog-to-digital conversion. Such a check is performed by comparing the input signals with inputs recalculated from the output, and signalling an error if disagreement occurs.

(3) Certain forms of output redundancy which are intricate or inconspicuous from a switching viewpoint may be

> easily checked when viewed as a system data output, usually as a function of time. Thus, the magnitude of the time-derivative of a succession of data values representing a slowly changing physical quantity may be checked to detect certain types of faults in the system which generates these values. Similarly, data word and block structure and the location of special symbols and signs within words can be verified to detect a limited family of faults.

A number of questions alluded to or raised in the preceding sections would merit further attention in a more detailed or specialized study of fault-detecting redundancy. For example, it may be possible to derive some generally valid rules on just how redundant *input* variables may be used to simplify both the irredundant and redundant portions of one network in a cascade of networks to be checked. Also, more guidance on the forms of preferred irredundant realizations of a given switching function or functions would be welcome. Similarly, estimates on the cost of the checking network, and on the change in this cost as a function of small changes in the form of the network to be checked, would be helpful to the designer in arriving at an economical, final, redundant network. Along the same line, we have not seriously attempted to answer the question, "What class of switching functions lead to the lowest redundancy ratios for fault detection"?

One theoretical question of limited practical importance was raised in Sec. II-C: When, if ever, can a realization of the complement $\bar{f}$ of a given unate function $f(x_1, x_2, \ldots x_n)$ be simplified by excluding the fundamental product $\bar{x}_1 \bar{x}_2 \ldots \bar{x}_n$? (A unate switching function is one which can be written with no complemented literals.)

By far the most challenging and important problems in the synthesis of redundant switching circuits lie in the sequential area, however. These problems are presently under investigation.

REFERENCES

1. J. von Neumann, "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," in *Automata Studies*, pp. 43-98, C. E. Shannon & J. McCarthy, eds., Annals of Math. Studies No. 34, (Princeton University Press, 1956).

2. E. F. Moore, and C. E. Shannon, "Reliable Circuits Using Less Reliable Relays," *J. of the Franklin Institute* **261** (9 and 10), pp. 191-208 and 281-297 (September, October 1956).

3. E. J. McCluskey, Jr., "Minimization of Boolean Functions," *Bell System Tech. J.* **35**, 6, pp. 1417-1444 (November 1956).

4. D. E. Muller, "Application of Boolean Algebra to Switching Circuit Design and to Error Detection," *IRE Trans.*, PGEC-3, 3, pp. 6-12 (September 1954).

5. R. W. Hamming, "Error Detecting and Error Correcting Codes," *Bell System Tech. J.* **29**, 9, pp. 147-160 (April 1950)

6. W. H. Kautz, "The Realization of Symmetric Switching Functions with Linear-Input Logical Elements," *IRE Trans.* PGEC, forthcoming.

7. W. W. Peterson, *Error Correcting Codes* (Wiley and Sons, Inc., New York, 1961).

8. E. J. McCluskey, Jr., "Comparison of Sequential and Iterative Circuits," AIEE Communication and Electronics **46**, pp. 1039-43 (January 1960).

STANFORD
RESEARCH
INSTITUTE

MENLO PARK, CALIFORNIA

REGIONAL OFFICES AND LABORATORIES

SOUTHERN CALIFORNIA LABORATORIES
820 Mission Street
South Pasadena, California

NEW YORK OFFICE
270 Park Avenue
New York 17, N. Y.

WASHINGTON OFFICE
711 14th Street, N. W.
Washington 5, D. C.

EUROPEAN OFFICE
Pelikanstrasse 37
Zurich 1, Switzerland

RESEARCH REPRESENTATIVES

HONOLULU, HAWAII
Finance Factors Building
195 South King Street
Honolulu, Hawaii

PORTLAND, OREGON
Suite 914, Equitable Building
421 Southwest 6th Avenue
Portland, Oregon

PHOENIX, ARIZONA
Suite 216, Central Plaza
3424 North Central Avenue
Phoenix, Arizona